L3: Entry 1 of 1 File: USPT May 7, 2002

DOCUMENT-IDENTIFIER: US 6383150 B1
TITLE: Network-based system for diagnosing balance disorders

Brief Summary Text (24):
In another aspect, the present invention provides a process for diagnosing the balance system of a patient, the process comprising the steps of a primary care physician of the patient referring the patient to the balance disorder diagnostic provider, the patient registering with the provider and completing a questionnaire for the provider; the provider obtaining authorization from an insurance carrier of the patient to administer the diagnostic balance tests; the patient scheduling an appointment for a plurality of diagnostic balance tests; an affiliate of the provider administering, at the appointed time, the plurality of diagnostic balance tests on the patient using diagnostic testing machines that generate diagnostic data, said plurality of diagnostic balance tests being administered at a first location; a network transmitting the diagnostic data from said diagnostic testing machines to a patient database that stores said diagnostic data; the provider retrieving, at a second location distinct and independent from the first location, the diagnostic data from said patient database; the provider evaluating the diagnostic data to diagnose the patient; the provider recommending therapy to improve the patient's vestibular functioning; the provider forwarding the diagnosis and recommended therapy to the primary care physician; and the primary care physician referring the patient to the clinic for therapy.

Brief Summary Text (26):
One of the advantages of the present invention is that it gives patients, healthcare providers, athletes, coaches, physical trainers, and others a more complete picture of how well the vestibular system is interacting with the body and brain. Another advantage is that it aids in accurate evaluation and treatment of the specific sensory abnormality by assisting the healthcare provider in distinguishing the source of the symptoms. A further advantage of the present information is that it promotes a team approach to working with patients and athletes, by coordinating athletic trainers with physicians and therapists.

Detailed Description Text (7):
FIG. 1, of course, is an exemplary embodiment of the present invention that incorporates many features not intended to be limiting. Numerous variations, enhancements, subtractions, and substitutions could be made to components of the system 100 without departing from several of the aspects of the present invention. For example, the global patient database 150 may reside at a regional balance center 160, or different kinds of patient information could be stored at different locations. Alternatively (or in addition), the regional balance center 160 may be substituted with an individual clinician qualified to evaluate vestibular balance data from any computer terminal connected to the network 140. Similarly, the balance testing center 110 may be replaced with a portable balance testing system. Also, the balance testing center 110 could employ different vestibular diagnostic tests, or transmit the results directly to the network 140 without first being collected and organized by a test data aggregator 130 such as a database. Furthermore, access could be denied altogether to patients 170 or primary care

physicians 180 without sacrificing all of the aspects of the present invention.

CLAIMS:

26. The system of claim 24, wherein said patient database comprises patient condition data, patient history data, patient test data, patient insurance data, patient billing data, and scheduling data, which the patient, a healthcare provider or an office administrator can access from the computer network.

27. A process for diagnosing the balance system of a patient, the process comprising:

the patient scheduling an appointment for a plurality of diagnostic balance tests;

administering, at the appointed time, the plurality of diagnostic balance tests on the patient using diagnostic testing machines that generate diagnostic data, said plurality of diagnostic balance tests being administered at a first location;

transmitting the diagnostic data from said diagnostic testing machines to a patient database that stores said diagnostic data;

retrieving, at a second location distinct and independent from the first location, the diagnostic data from said patient database; and

evaluating the diagnostic data to diagnose the patient.

Previous Doc Next Doc Go to Doc#

L12: Entry 2 of 2 File: USPT Dec 21, 1999

DOCUMENT-IDENTIFIER: US 6004276 A
TITLE: Open architecture cardiology information system

Abstract Text (1):
A clinical information reporting system for use with an electronic database for a health care facility, the electronic database being a rotational and modular database for the provision of a scalable and extensible configuration preferably consisting of a workstation as the base configuration and being configurable for use in small and medium network situations and being particularly adapted for the receipt, manipulation, modification and generation of cardiology reports such as resting ECG records and stress ECG records.

Brief Summary Text (3):
The present invention is directed to an improved object oriented information system for use in a hospital and more particularly for use by the cardiology and administrative departments of a hospital.

Brief Summary Text (6):
The medical records of a patient in a hospital typically contain laboratory and test data, physicians orders and other information which is important to the treatment of the patient and which is also typically not available on the HIS. Additionally, with the increase in insurance and other third party payment plans, it is important for a hospital to accurately bill for the services and treatment provided as well as to monitor their costs of providing the treatment and services. In many hospitals which provide the full range of cardiac services, the hospital may receive between 25 and 35 percent of their revenues from cardiology patients. This category of patients also has one of the highest needs for rapid and complete access to their medical records.

Brief Summary Text (7):
In about 1976, the Marquette Electronics Company introduced a computerized electrocardiography management system. This system provided data storage, viewing, retrieval and report generation capabilities for diagnostic 12 lead electrocardiographs acquired from Marquette equipment. This type of system is known as a closed system because it communicates only with electrocardiography equipment purchased from the manufacturer of the management system. Therefore, once a hospital purchases a closed management system, all future pieces of medical equipment must be purchased from the same manufacturer in order to communicate with the management system. With the increased concerns about the increasing costs of healthcare, it is increasingly important that each piece of equipment purchased by a hospital or clinic communicate with existing equipment and perform as many functions as possible.

Brief Summary Text (8):
ECG management systems are a vital component of the computerized ECG equipment market. These systems expedite the flow of ECG reports in a hospital by improving the access to records for copies and serial comparison analysis. The computer also assists with the information routing tasks such as editing, sorting, tracking and

printing of the ECG records. Many of the functions of a cardiology department are significantly improved by the increased access to the ECG information. The staffing required to process ECGs may be reduced with the addition of an ECG management system.

Brief Summary Text (9):
The cardiology diagnostic department of a hospital uses an ECG management system extensively. One part of an ECG management system is typically a computer based ECG interpretation program. Although this program is not as skilled as a cardiologist, the program often provides a useful initial review from which the cardiologist may make further revisions and provide the final diagnosis.

Brief Summary Text (11):
Some of the basic functions of the current ECG management systems are data storage, data retrieval, data viewing and the streamlining of the overread process. In the prior practice which used paper copies of the ECG of a patient's records or with an ECG management system, an unconfirmed report of the ECG test is provided to the central station for later overreading or review by a cardiologist. A second copy of the unconfirmed report is typically left in the patient's record at the nurse's station. At some point, the physician on duty will pick up the ECG tracings from the central station and overread or edit them as time permits. The annotated reports are then returned to the central station for editing and data entry. Once the edited record is entered, the confirmed report is then printed and dispatched to the nurse's station to replace the unconfirmed report in the patient's record. It is extremely important that the patient's records not be lost or delayed. The major advantage of the ECG management system is that the transfer to the patient's primary record is instantaneous once it has been entered and there is no likelihood that the confirmed report will be lost or delayed in the hospital delivery system. Additionally, there is less opportunity for data entry errors because it is no longer necessary to clerically enter the physician's comments or diagnosis.

Brief Summary Text (13):
Additionally, smaller hospitals, clinics or cardiology offices may contract with outside services for data storage and/or overread services. The ECG management system provides the smaller facility with the benefit of an ECG management system without the additional investment or additional staff. The administrator of the facility may also obtain traffic and management information to help their facility to be more cost effective and efficient.

Brief Summary Text (14):
The currently available ECG management systems have only touched the surface of potential applications for a cardiology information system (CIS). This inability to reach their full potential has resulted primarily from the use of closed systems which limit their own usefulness to the breadth of the product line offered by their manufacturer. As a result of the existence of "closed" systems, a number of software development companies have begun selling "translation boxes" to hospitals to enable the various acquisition devices to communicate with the pre-existing hospital systems.

Brief Summary Text (16):
In addition to the record management benefits described above, a cardiology patient typically has other procedures and records which must be managed and archived. For example, the cardiology patient may have HOLTER records, stress test records, catheterization laboratory records, echocardiographic records, electrophysiology, telemetry, metabolic testing records or pacemaker follow-up test records. At present, only a few of these records are accessible to a hospital through the current ECG management system.

Brief Summary Text (17):
Therefore, there is a need for a cardiology information system which provides

individual and integrated procedure reports that incorporate key clinical data from all available procedures to reliably and accurately provide proper clinical decision making and accurate reimbursement.

Brief Summary Text (18):
Additionally, there is a need for a cardiology system that provides a simple graphic user interface and standard PC hardware which also uses other standard PC software for word processing, spreadsheet and desktop publishing applications.

Brief Summary Text (19):
There is yet another need is for a cardiology information system which provides a standardized hospital information system connection so that common patient census data, billing capture, results reporting and order driven systems may be used throughout the hospital.

Brief Summary Text (20):
There is a further need for a cardiology information system which provides a standardized communication protocol so that common patient data, billing information, procedure results and medical records information may be acquired by nearly any currently available data acquisition device which may then be reviewed throughout the hospital.

Brief Summary Text (22):
The present invention is directed to an open architecture cardiology information system which preferably has modular object oriented software to allow for easy expansion, relational database management, custom reporting, local and wide area networks and communication with equipment from a variety of manufacturers and which is readily expandable for use in cardiology group clinics as well as in small, medium and large hospitals.

Brief Summary Text (23):
The cardiology information system of the present invention preferably includes resting and exercise ECG modules; procedural management modules; administrative reporting modules; interfaces with other manufacturers equipment and is preferably a MICROSOFT SOLUTION PROVIDER product. Additionally, the present invention provides modules which allow for the communication with various cardiac catheterization laboratory systems and electrophysiology systems and includes a HIS connection. Furthermore, it is anticipated that the present system may be expanded to include modules which allow for the communication with systems that perform cardiac imaging, Holter monitoring, telemetry, echocardiography, stress echo cardiography and pacer detection as well as administrative functions such as procedure coding, scheduling, inventory management, outcomes management and custom reporting. This is accomplished by the modular nature of the present invention and the provision of a versatile shell operating module which also allows for the use by multiple users to perform multiple tasks including word processing, integrated spread sheets and the storage of other data. The framework provides the basic building blocks or classes that may be used by workstation products to implement the desired records, fields or data repositories. The framework does not implement any records but does provide abstract classes which are used by the workstation products to implement the records. In the present invention, the workstation functions as the basic fundamental operating unit of the system. The workstation enables the user to review, edit and store various physiological signals acquired from physiological signal acquisition devices in combination with other patient related data and patient information.

Brief Summary Text (25):
In the present invention, the cardiology information system design generally includes software modules for framework shell modules; framework applet modules;

Brief Summary Text (28):

The services layer includes modules for an event logger applet and an access services applet. The rendering layer includes modules for a record presentation applet and an applet widgets applet. The manipulation layer includes modules for a field framework applet and a record framework applet. The access layer includes a module for a database access applet. The dynamically-loadable framework layer includes modules for administrative reports applets;

Brief Summary Text (30):
The modules of the workstation products framework preferably run together as a single process under a defined software operating system such as WINDOWS NT. The shell module is the sole executable module of the software operating system process. The remaining modules are operating system dynamic load libraries. Within this single process, all modules run as a single operating system thread, although it is anticipated that additional threads may be used such as for requests made to persistent storage by a database access module.

Drawing Description Text (44):
FIG. 43 is a diagrammatic view, illustrating the database accessor view of the recordset accessor field framework of the present invention where the arrows show the calls made in the direction of the calls;

Drawing Description Text (45):
FIG. 44 is a diagrammatic view illustrating the recordset view of the recordset database field framework of the present invention where the white arrows show the calls made in the direction of the calls and the black arrows show the flow of data;

Drawing Description Text (48):
FIG. 47 is a diagrammatic view illustrating the database communication processes of the client/server of the present invention;

Drawing Description Text (86):
FIG. 85 is a diagrammatic view of an overview of the Database Access classes of the present invention;

Drawing Description Text (87):
FIG. 86 is a diagrammatic view of the Database Access interactions with the Field Framework of the present invention;

Drawing Description Text (88):
FIG. 87 is a diagrammatic view of the Database Access interactions with the Record Framework of the present invention;

Drawing Description Text (105):
FIG. 101 is a diagrammatic view illustrating the Record Framework interactions with the Database in the present invention;

Detailed Description Text (2):
The cardiology information system (CIS) is a member of the family of workstation systems. It is a computer-based cardiology data management system with controlled security access. An important purpose of the CIS is to provide a means to establish, maintain and access organized electronic storage of complete cardiology patient records. The CIS will provide for several configurations, ranging from a single computer to a dedicated server for a large computer network supporting a large number of hard-wired and modem-access PCs.

Detailed Description Text (11):
As shown in FIGS. 2 and 3, the CIS system is preferably partitioned into two major subsystems, the CIS platform and the CIS application software. The CIS platform represents the hardware and the support software which is used to run the CIS

application software. It includes the client hardware and client operating system, server hardware and server operating system, network hardware and network operating system and peripheral devices and device drivers. The CIS application software is the software which has been developed to implement the product-specific CIS requirements. The CIS application software includes the client and server application software and third-party software (other than Platform software) used to satisfy system requirements. The interface between the two CIS subsystems, Platform and Application Software, is preferably defined by a WIN32 API which is the Windows NT Application Interface and an ODBC which is a generic Database query language.

Detailed Description Text (19):
The ability to allow a user to log in to the system from a remote site may also be supported in the preferred form of the present invention using the Windows NT "Remote Access Service" (RAS). The RAS allows a user to connect to the network from a remote site and work with the CIS application as though they were physically in the office, directly connected to the network or local database (although network access to a remote workstation will be slower). The system preferably allows three types of remote connection. A modem on the remote Client may be connected across standard telephone lines to a modem on the system network. Standard modem data rates up to about 28.8 kbits/sec are supported. The ISDN mode is also supported because this mode uses dedicated phone lines to support high speed data links (up to 128 kbits/sec). Because ISDN requires support from a local telephone carrier; it is not universally available, but most major metropolitan areas have it. This solution would serve well for a physicians' office connecting to a local hospital CIS system. The third type of remote connection is X.25. This is desirable because it is an international standard used for exchanging data across public telephone networks and it may use leased lines, if available, for fast local connections.

Detailed Description Text (20):
The use of the Windows NT Server in the preferred form of the present invention provides a domain-based naming and logon system. A domain is an aggregation of workstations for administrative purposes. Multiple servers within a single domain can share a user database and can be administered from a single workstation. Multiple domains may be established, each with its own user database; "trust relationships" may be created between domains which allow selected users from one domain to have access to a different domain. Both single domain and multiple domain configurations are supported by CIS. In the present invention, in a single domain/single server system, all CIS users are preferably maintained by the single server and all patient and procedure records are stored by the single server as shown in FIG. 6. In a single domain/multiple server scenario as shown in FIG. 7, all CIS users also preferably share a common user database (which is replicated on all servers). Conceptually, a user logs on to the domain, rather than to a particular server and a user selects one of the servers as the primary CIS server; this primary server is then the default for running queries on the CIS database. A user can access data on any other server to which the user has rights. In the scenario shown in FIG. 7, all EKG procedure records reside on one server and all the Stress procedure records reside on the other; patient information records would reside on both, with duplication for patients with both procedures. As shown in FIG. 8, with multiple domains, separate user databases are maintained for each domain. With multiple domains, each procedure record type may reside on its respective server and patient information records may be stored on both servers. A "trust relationship" is preferably established to give users in one domain access to data in the other domain. For example, the group `Stress Tech` in the Stress domain might be given rights to read EKG records from the EKG domain.

Detailed Description Text (22):
Because of the various system configurations supported in the present invention, the client and server subsystems are not independent. Therefore, these two subsystems are combined and referred to herein as "workstations". A workstation

serves as the user's interface to the CIS application software. There are three CIS workstation types used in various combinations to address the three CIS system configurations in the preferred form of the present invention. Each of these workstation types are preferably a PC capable of running a Windows NT software program. The client workstation preferably runs the CIS client software and is used in both the small network and medium network models. The software is preferably Microsoft Windows NT workstation software. The server workstation is preferably used to run the network operating system and CIS database server software. Depending on the individual system configuration, the server workstation may also provide one or more network services including printer and modem sharing, digital scanner services, client data backup service, etc. This workstation is preferably used only in the medium network model and the software is preferably Microsoft Windows NT Server with a Microsoft SQL-Server.

Detailed Description Text (28):
As discussed briefly above and as discussed in more detail below, the software design of the present invention is preferably based on an object oriented software design to provide the present invention with the flexibility necessary to accommodate upgrades and scalability. The design philosophy is generally based on a three-tiered Donovan model which enables the CIS application to "plug and play" with different components for each of the three tiers while the system requirements change. This approach confines the modification impacts to a specific tier and therefore; increases the reusability of the other tiers. With this approach, the top tier provides a mechanism for the user to interact with the application. This top tier provides the invocation and navigation functionality of the system. Various interfaces communicate between the top tier and the second tier. The second tier preferably provides the business logic and decision making infrastructure for the system. This middle tier also provides the translations and controls between the top tier and bottom tier. Various other interfaces communicate between the second tier and the third tier. The third tier is somewhat protected from the user interactions with the top tier and provides the basic control, management and integrity services for the data supported by the application. The software of the CIS also preferably employs various industry standards for portability such as OLE and ODBC programming languages. The philosophy of the software design places a great deal of emphasis on identifying and designing the system based on the stable elements of the system, such as patient information, test procedures and users. The volatile design elements, such as operating system specific or DBMS related functions are isolated with a wrapper or interface so that the application may be readily moved to different operating systems or to accommodate software operational related changes. Additionally, common mechanisms or common protocols have also been designed for functions that have a common way of operating or communicating. For example, as discussed more fully below, the adoption of SCP+ as a communication standard for communication. This allows the CIS application software to use one protocol to communicate with all external acquisition devices or instruments and any necessary data translation is performed external to the core of the CIS. The benefit of this approach is that the CIS framework does not need to know the data format details of each acquisition instrument, and both the CIS and instrument specific software become self contained modules.

Detailed Description Text (30):
A further advantage of this design approach is that the commonly shared data and functions are centralized so that duplicative functionality is minimized. The centralization is not related to the physical location of the data and functions, but relates more to the operational functionality of the data and functions. In the preferred form of this invention, only one implementation of a function exists within the application. This approach reduces the number of routines and files which need to be modified and updated and therefore, the software is more maintainable. Similarly, this approach uses and controls constrained resources wisely by eliminating unnecessary network data exchanges by keeping the data and processes lose to where the action occurs. This is particularly important for the

database and network resources which are typically overburdened.

Detailed Description Text (31):
The CIS supports various system configurations, including a stand-alone configuration in which, the client and server run on the same hardware platform (workstation) and may or may not be a node on a larger network. In this stand alone configuration, the scope of the system is limited to the physical resources of the dedicated workstation, and any peripheral devices configured for use by the client or server as described more fully below. The CIS will also support a Networked, Isolated Database referred to herein as a small network. In this configuration, the server platform is a node on a network. Each properly configured virtual network node, including the server, may be allowed access to the server as a system client. In the small network configuration, the scope of the system is limited to the physical resources of all properly configured network nodes, and any peripheral devices configured for use by client or server applications on the nodes as described more fully below. Small networks preferably employ at least a 4 MB/sec. topology (or the equivalent) and allows no more than about twelve local and remote clients concurrently (use of the server as a client counts as one of the twelve clients).

Detailed Description Text (34):
The CIS system functionally operates as one or more client applications attached to one database server at a time. In the preferred form of the present invention, the CIS system is a workstation based product. Requirements to support additional functionality provided by the CIS product are described more fully below. For example, the CIS product allows the user to select a record in the list using a bar-code scanner and if a bar-code scanner is used to select the record, the record is automatically opened for review. The CIS product also allows the user to enter patient data using the bar-code scanner so that fields represented by the bar code such as MRN and patient name would be automatically filled in. The CIS product also allows various communication paths, interface protocols, and operational scenarios to be supported by the workstation.

Detailed Description Text (36):
As discussed more fully below, If the user has installed the optional Resting ECG Reports Application, the user is also able to perform a number of functions, including those described below, for all Resting ECG records in the CIS database. The optional Resting ECG Reports Application provides the user with report review, serial presentation report review, record editing, record abridgment, report distribution and notification, report printing and resting ECG report setup features. Also as discussed more fully below, if the user has installed the optional Stress ECG Reports Application, the user will be able to perform a number of functions, including those listed below, for all Stress ECG records in the CIS database. For example, the user will be able to perform report review, serial presentation report review, record editing, record abridgment, report distribution and notification, report printing and stress ECG report setup.

Detailed Description Text (38):
In the preferred form of the present invention, the Resting ECG Reports and Stress ECG Reports are available to the customer as system options, and the CIS is designed to readily include additional system functions to be implemented on the CIS. These additional options include, an allow networked, integrated database option, touchscreen overlay, billing notification, CPT Coding and Management, Inventory Control, Scheduling, Advanced Database Query, HIS Interface Configuration, E-mail, Fax Polling, Merged Reports,Cath-lab Procedure Reports, Electrophysiology Reports, Imaging Reports, Holter Reports, Rehabilitation ECG Reports, Pulmonary Reports, Resting ECG Analysis, Advanced Scanned and Faxed Report Analysis, Tool Bar customization, Work Forwarding, Acronym Macros, Auto-Lead-Size Sensing and Outcomes Reporting.

Detailed Description Text (39):
As mentioned briefly above, the CIS includes workstation systems that are made up of one or more PC-based workstations connected together, accessing one or more databases. Each workstation provides the user with a set of functions that will perform data processing, and in some cases data acquisition. Multiple functions can be open on a workstation at one time. Users at separate workstations are able to use the same functions and view the same data without interfering with each other (although only one user at a time will be able to edit a particular record). The workstation shell provides the basic platform from which all other workstation functions operate. Access to the shell is secured. The workstation shell essentially operates as a "Program Manager." Once properly logged onto the shell, the user has access to various standard features, such as viewing customized lists of patient/procedure records on the database, system setup, system administration, including user/group setup, backup, and archive, patient data and demographics entry and management, administrative reports generation and review and scheduled and on-demand transfer of patient and procedure records to/from other systems, including download of patient information records to instruments. In addition to the standard features, products based on the Workstation may add additional features, such as report generation and editing, data acquisition, and inventory. Functions such as data transfer may be performed in the background while the user is performing other tasks such as reviewing reports. FIG. 9 shows the interface between system functions within each workstation.

Detailed Description Text (41):
The workstation systems are preferably capable of supporting the basic configurations briefly mentioned above. FIG. 10 shows the physical structure of a stand-alone workstation. In this product configuration, the system consists of a single workstation and a local database is maintained within the workstation. The system setup information for this configuration is maintained on the workstation. Communication with any other systems is preferably via the data transfer function described below.

Detailed Description Text (42):
FIG. 11 shows the physical structure of workstations in a networked product configuration featuring an isolated database (small network). In this product configuration, the system preferably consists of one or more workstations and one or more database servers, networked. This configuration supports database server configurations which are dedicated database servers and/or database servers resident on client workstations. The system setup information for this configuration is preferably maintained in a single location on the network and therefore all workstations on the network use the same system setup regardless of which database server they are attached to. In a multi-database environment, such as the medium network configuration shown in FIG. 12, the user may view the list of databases and query any of the databases from any workstation on the network.

Detailed Description Text (43):
In the present invention, database queries associated with viewing lists of patient/procedure records may operate on only a single database at a time, and database queries associated with administrative reports are able to operate on all databases on the network. The results of cross database queries are typically reconciled in this configuration. For instance, if 30 patients are stored on database A and 50 patients are stored on database B, but seven of those entries represent the same patient (seen at both hospitals), a query requesting the number of patients seen would return 80, even though only 73 were actually seen. Finally, the multi-database user is preferably able to copy data from one database to another easily (such as using "drag and drop") and communication with any other systems not connected to the network shall be via the data transfer function described below. Common network resources will typically include optional printers, fax routers, modem routers, scanners, etc.

Detailed Description Text (44):
The physical structure of the workstations of the present invention in a networked product configuration featuring integrated database (medium network) is the same as that for isolated databases (small network). In the medium network product configuration, the CIS consists of one or more workstations and two or more database servers, networked. Dedicated database servers and/or database servers resident on client workstations are supported. The system setup information for this configuration is maintained in a single location on the network and therefore all workstations on the network use the same system setup regardless of which database server they are attached to. In a multi-database environment as contemplated by this configuration, the user is able to configure the presentation to view the data as a single logical database or use separately visible databases. The user is also able to query any of the databases and the queries are operable on multiple databases at a time. the user may also copy data from one database to another via "drag and drop". In this configuration, communication with any other systems not connected to the network is via the data transfer function described below.

Detailed Description Text (45):
Access to the CIS may occur via the initiation of the workstation application from the program manager (or its equivalent) on a local client workstation. The user may also initiate the workstation application from the program manager (or its equivalent) on a remote client workstation. The system administrator may also access the database server and third-party software packages are able to access the database via mechanisms supplied by the database. All access to the system is secured so that a valid user name and password is required. The system provides security to ensure that data is accessed only by authorized users. The system administrator may setup a default user to support automatic logins and an error will logged in the event log for each failed login attempt. The system administrator is able to select a maximum number of allowed login retries and if the number of consecutive failed user login attempts exceeds the maximum number of login retries, logins shall be suspended at that workstation for ten minutes. Once the user has gained access to the workstation application via one of these methods, the workstation shell will allow the user to access the functionality described below as well as product specific functionality delineated in the appropriate product specification. Finally, only one local client is allowed to run on each system node.

Detailed Description Text (46):
The system supports both background and foreground processing. Background processing (such as scheduled record transfer or administrative report generation) is allowed to proceed as required whenever a database server is active and does not require the presence of any logged in users. Foreground processing occurs in response to user commands from an active workstation. All client applications running on the system are able to simultaneously execute the same client functions subject to the user privilege levels, administrative user restrictions, and data integrity requirements as described below.

Detailed Description Text (47):
Only one client at a time is preferably allowed to edit an element of the database. Other clients will not be prevented from viewing data being edited by another client, but their view of the data will indicate that the data is locked for editing. The system will also not allow data in use by multiple clients to be deleted. The system will also preferably prevent loss or corruption of data due to system crashes or power failures. The workstation shell provides the user access to the system functions described below.

Detailed Description Text (48):
The user is able to view lists of the patient/procedure records contained on the database and is able to use standard and user-customized list filters to control

what information is displayed. Standard filters include, all patients, all procedures, all conflicting procedures, deleted records and received scans/faxes. The user is also able to control the presentation of the list (column content, column order, and sort order) by sorting the list and printing the list. The user may also select records in the list and initiate their review (the procedure report functional specifications provide the requirements for review) using the keyboard or a mouse. The user may initiate creation of a new record and associate faxed/scanned records with new or existing patients.

Detailed Description Text (51):
Various communication paths, interface protocols, and operational scenarios are provided by the system for transferring patient information and procedure records to/from the Workstations. The system supports record transmission and receipt for on-demand record transmission, including target selection, transfer timing, and record selection, requested and unsolicited record receipt from permitted sources and scheduled record transmission management, including target selection and transfer timing. The system also provides transfer status information to the user on request.

Detailed Description Text (54):
The system response time to user functions requiring access to the system database is preferably less than three seconds, and system response time to user functions not requiring access to the system database are preferably less than about 500 ms. The preferred system requirements for system response time to user functions do not apply while system backup or archive operations are in process and do not apply to workstations engaged in data transfer operations. The preferred requirements for system response time to user functions also do not apply to a specific workstation if minimum memory availability requirements are not met on that workstation. On a Hewlett-Packard LaserJet IIIsi brand printer, text-only documents preferably print at a minimum rate of 15 pages per minute, and mixed text and graphics documents will print at a minimum rate of four pages per minute. The mixed text and waveform documents also print at a minimum rate of three pages per minute, and the user is able to establish priority queues for system printers. Performance requirements for printing preferably only apply to top-priority queues. These requirements do not apply while system backup, archive or data transfer operations are in process or if minimum memory availability requirements cannot be met.

Detailed Description Text (56):
In the preferred form of the present invention, workstation systems are made up of one or more PC-based workstations connected together accessing one or more databases as generally illustrated in FIGS. 10-12. Each workstation forms the basic functionality of the present invention and will provide the user with a set of functions that will perform data processing and, in some cases, data acquisition. Users at separate workstations are able to use the same functions and view the same data simultaneously without interfering with each other. The shell function of the present invention provides the basic platform from which all other workstation functions operate. The shell provides basic system functionality (including viewing lists of patient and procedure records and launching appropriate functions based on user commands). FIG. 13 shows the interface between the shell function and the other functions of the present invention. Upon startup of the application, the CIS logo and product copyright are displayed. The shell function determines which workstation functional entities are present and available, and diagnostics are performed on each available functional entity to determine operability. If any functional entity fails the diagnostics, the user is notified of the failure. If any functional entity fails the diagnostics, the user may be prompted to exit to the application or continue with limited functionality. The database contains a patient folder for each unique patient. Each patient folder includes a current demographics record which represents the most recent patient demographics information for the patient and zero or more procedure records associated with the patient. Each procedure record contains demographic data representing the patient's

information at the time the procedure occurred.

Detailed Description Text (57):
The user is able to view a variety of lists of patient/procedure records resident on the database. The user may view various types of lists, including the patient list which consists of a list of all patient folders that meet the filter criteria, the procedure list which consists of a list of all procedures that meet the filter criteria and the patient folder which consists of a list of all of a specific patient's procedures that meet the filter criteria. Each list is associated with a filter allowing the user to limit what database entries will be included in the list, and how the data will be presented. The user is also able to display the lists by selecting a list filter or by using the "find" control. User access to specific records may be limited by the system administrator by selecting various privilege levels which function to limit access to records by filtering the availability to data and records. For example, if user JOE is only allowed to access group filters, and the only filters available for his group are RESTING records with attending physician of DR. CARL or DR. DRAB, other patient records are protected from JOE.

Detailed Description Text (58):
The user may display an initial list of patient information or records by selecting from a list of filters. The list of filters available for selection varies based on what filters have been created and what the user has access to. A patient filter, procedure filters, conflicting records filter, scanned or faxed images filter and deleted records filter are initially provided with the system. The patient filter list includes all patients, without displaying any of the associated procedures. The procedure filters include a list of all procedures on the database, a procedure list for each available procedure and a list for unconfirmed reports for specific procedure. The conflicting records filter includes all records that have been marked as conflicting. The faxed and/or scanned images filter includes all faxed/scanned images that have not been associated with a patient/procedure. The deleted records filter includes all records marked for deletion. Although the user is not able to change the filter criteria for these lists, the user is able to change the presentation of the lists.

Detailed Description Text (59):
The user of the CIS system is able to create a list by searching for records matching a specified criteria. For example, the user may choose the search fields for patient last name, MRN, social security number, or billing number. The user may also choose a search filter from the list of patient filters described above and the user is able to enter a string of characters to be matched. At the user's command, the system searches for any records in the database that meet the filter criteria and contain a value in the specified field that wholly or partially matches the input string. If multiple matching records are found, a patient list is displayed showing all patient folders containing matching records. If only a single matching record is found, the patient folder containing the record will be displayed with the matching record(s) highlighted. If no matching records are found, the user is so notified.

Detailed Description Text (60):
The following preferred system requirements identify the preferred method of how a list will be displayed. Entries in a patient list will represent patient folders of patients in the database meeting the list criteria. Entries in a procedure list will represent procedure records in the database meeting the list criteria. Patient folder lists preferably consist of the specified patient's current demographics record and each of the patient's procedure records meeting the list criteria. The patient's name and MRN are displayed in the title bar of the window and because patient folders always originate from patient lists, the filter criteria is defined in the patient list filters. The conflicting records list displays a list of all records that are currently marked as conflicting. The faxed and/or scanned images

list displays a list of all images stored on the system that have not been assigned to a patient/procedure. This list displays the date/time image was received and image file size for each image. The deleted records list displays a list of all records that are currently marked for deletion. The column setup associated with the selected filter will dictate what data is displayed for each entry. The displayed list is initially sorted based on the field in the left most column in the order (ascending/descending) selected in the filter. If a record has been marked as conflicting (the system was unable to determine whether the record matched another record), the list will indicate the conflict via a visual cue. FIGS. 14A-C illustrate examples of some of the possible list views with the present invention.

Detailed Description Text (61):
The user can edit existing filters or create a new filter. A filter is defined by initially selecting a list type, selecting the criteria used to determine which entries will be included, selecting which columns of data will be included in the view and activating the filter (which may or may not include saving). List filters are defined by selecting either the patient list or procedure list. The filter criteria is selected by selecting qualifying fields to selectively view a subset of the entries on the database. In the present system, the user may only change the filter criteria for the patient, procedure, and patient folder lists. The user may not change the criteria for the conflicting records, faxed and/or scanned images, and deleted records lists. Although it is obvious why both patient and procedure data is used in procedure lists, it is not so obvious why procedure data is used in patient lists. The reason is twofold: 1) the user may want to view "All patients with unconfirmed rest records;" and 2) the user may open a patient folder (containing procedure records) from the patient list; and, therefore, the user must have filter criteria for the patient folder.

Detailed Description Text (69):
The shell function of the workstation product allows the user to create a new patient folder by adding a new current demographics record to the database. For each type of procedure function available on the system, the shell function will allow the user to add a new procedure record to the database. The user is required to select an existing patient folder to associate with the new record or indicate that the procedure is for a new patient.

Detailed Description Text (70):
The user may select one or more entries in the list, and the system of the present invention supports both contiguous and disjoint selections. Once one or more entries are selected, the user is able to initiate the actions described below. The user shall be able to initiate printing of the selected entry(s). If the entry represents a current demographics record, the printing without viewing requirements of the patient demographics function will apply. If the entry represents a procedure record, the printing without viewing requirements in the applicable procedure reports function will apply. If the entry represents a patient folder, the patient summary report containing the current patient address, the patient information fields enabled for viewing in the list filter and each of the patient's procedures that meet the filter criteria, are printed as defined in the filter's column setup with one procedure per line. If the entry represents an image, the user may print the image as a BLOb (Binary Large Object). If the image has been assigned to a patient/procedure, the attached information will also be printed.

Detailed Description Text (73):
The user is also allowed to perform editing of the displayed fields of the conflicting record to resolve the conflict, to accept the conflicting record as a new patient if the record's MRN is unique within the MRN model for the record's site, to accept the conflicting record as an alias for the patient with which it conflicts if the MRN/MRN model and date of birth match, but name or gender are different, or if the MRN model is different (same patient, different hospital using

different MRN model) to resolve record conflicts. If the conflict is resolved (and no new conflict is generated), the conflict marking is removed from the record. The user may mark the priority of a record by marking a procedure record as STAT or Normal. STAT indicates that a record needs immediate attention. The user may delete selected entries but is unable to delete records that are currently being viewed or edited by another user. The user is also required to verify deletions of records. If the selected entry is a patient folder or a current demographics record, all associated procedure records will also be deleted. The user may not be allowed to delete archived entries. As a safeguard, it is important to note that records are not really deleted from the database until an authorized user instigates a purge. Before the purge is instigated records are only marked as deleted, which removes them from any lists that don't include deleted records.

Detailed Description Text (74):
If the selected entry has a status of "deleted," the user may be able to restore the selected record(s) to the database (removing the deleted status) or purge the selected record(s) from the database. Once a record has been purged, it cannot be recovered. The shell function specifically requires user verification of the purge function.

Detailed Description Text (81):
In addition to the above described features, it is anticipated that the Shell function may also incorporate an "Encounter" or grouping concept (where a single encounter may contain several different types of procedures). Additionally, a feature such as Soundex may be incorporated to the list search capabilities so that user may enter the approximate spelling of the name and the function will show close matches. A further feature such as ordering Physician information and procedure coding may be added to the list views options, and optical character recognition (OCR) features for faxed or scanned images may be incorporated into the CIS product. The shell function may also preferably determine if any data recovery activities need to be performed (because an unexpected termination of the application occurred, either through remote shutdown by the system administrator or through a crash or power down of the workstation). If data does need to be recovered, each record that was open for edit will be reviewed to determine if a database record matches the recovered record, and the user will be notified. If the records do not match and the database record has been edited since the termination, the user will be notified and the record from the database shall be opened in edit mode. Additionally, the recovered record may be opened in view mode, and the display will clearly show that this is a recovered copy only. The user may then manually compare the two versions and make the desired changes to the database record. If the records do not match and the database record has not been edited since the termination, the database record will be locked for editing and the changes retrieved. The user will then be notified of the recovery of the record. The list entries will also be sorted based on the column order and the left-most column will represent the primary sort key, the next column to the right will represent the secondary sort key, and so on. Less significant sort keys represent sorts done "within" the previous column, without disturbing the order of the more significant sorts. Patient name will preferably always be sorted based on last name, first name, then middle name. Sort depth of the present embodiment preferably does not exceed five levels. Use of the search function will also not result in deselection of current selections. If a patient information record or procedure record is currently being edited by another user, the list will preferably indicate via a visual cue that the record is being edited. A preferred form of the present invention will also include an automatic screen secure mechanism so that, if a different user returns, the user will be required to logout and login. The present invention also preferably allows the user to set duplicate procedure records to conflicting status, and the user will be allowed to replace the current procedure record with the conflicting record, change the association of the conflicting record to a different patient, or delete the conflicting record.

Detailed Description Text (82):
An integral component of the preferred form of the present invention relates to the transfer of data between the workstation component of the CIS and various physiological signal acquisition devices or instruments. The functional components for data transfer in the CIS includes the communication paths, interface protocols and operational scenarios which are supported for transferring patient information and procedure records to/from a workstation in accordance with the present invention. FIG. 16 shows the interface between the data transfer function and other devices. The data transfer function of the present invention supports several physical pathways for communication with external devices and instruments. For example, the data transfer may occur via RS-232 serial ports, RJ-11 interfaces, inter-network interfaces, floppy disk interfaces, and an SCSI interface. An RS-232 serial port is supported for the serial transfer of records. An RJ-11 interface is supported for modem and fax transfer of records. Record transfer between other compatible network systems will be supported. Record transfers to or from 51/4" and/or 31/2" floppy diskettes (in DOS format) is supported. A SCSI interface is also supported for receipt of digital images of records from external scanners.

Detailed Description Text (86):
As described briefly above, patient information records and procedure records may be transmitted to other instruments or networks. When a record is transmitted, the appropriate communication protocol to be used is determined from the Connections List. The data transfer function translates the record to the appropriate format for the transfer and sends the translated record, using retries or other error-correcting techniques as appropriate.

Detailed Description Text (87):
With the system of the present invention, patient information records and procedure records may also be received from other instruments or networks. When a record is received, the communication protocol to be used is be determined from the connections list. The data transfer function receives the record, using retries or other error correcting techniques as appropriate and verifies the integrity of the record. If no errors are detected, the data transfer function translates the record to the system database format and the data transfer function initiates the appropriate record workflow actions. If the device transferring the record requests a serial record transfer, the data transfer function transfers the most recent record (if any) associated with the same patient as the received record.

Detailed Description Text (98):
In addition to the features of the data transfer function described above, it is anticipated that the data transfer function may be enhanced to accept records from third-party devices so that those additional records may be stored in the system database format and manipulated. Additionally, the ability to automatically detect the protocol to use on unsolicited transfers is an anticipated enhancement. Similarly, the ability to perform external device initiated transfers by supporting unsolicited requests for transmitting data out of the system with some type of criteria/security requirement to make sure that data is only transmitted to approved sites is a desirable addition.

Detailed Description Text (108):
The site list may also include a technician list of up to about 250 technicians per site. For each technician, the system administrator is requested to supply the technician name, the unique technician number, the notification path and the department (from the department list defined above) with which the technician is associated (or "none"). The system administrator may also control the hardware configuration of the system utilizing standard system services provided by the operating system. This application uses the system time and date and resource list (local and network, including identification of clients and servers). If the system contains multiple databases, the system administrator will be able to define what information is stored on each database. One database may be designated as the

primary container of system data (all user-defined parameters specified in the functional specifications), and each site/procedure type combination is assigned to one of the databases. For example, in a procedure based organization, resting records from site A may be stored on the Resting Database; resting records from site B may stored on the Resting Database; stress records from site A may be stored on the Stress Database, and stress records from site B may be stored on the Stress Database. In a site based organization, the resting records from site A may be stored on Database A; the stress records from site A may be stored on Database A; the resting records from site B may be stored on Database B, and the stress records from site B may be stored on Database B.

Detailed Description Text (120):
The system administrator is able to display memory and disk utilization and/or fault status for all nodes on which a server or system client is active. The system administrator will also have limited access to standard network and database server maintenance and tuning features. The system administrator is also able to perform system backups and restores and may backup the system database, system applications and the system configuration. In the present invention, the system database includes the patient records and other data within the system database.

Detailed Description Text (121):
The system applications consist of the system software. The system configuration includes user lists, group definitions, system lists, report format definitions, etc. If more than one backup device is available, the system administrator is able to select the device on which the backup or restore is to occur. The system administrator may perform a total or incremental backup and may request that a manual backup be performed on the system database, system applications and/or the system configuration. The system administrator is able to establish times at which scheduled backups are to be performed based on days between incremental backups or total backups. The system will automatically perform scheduled backups at the scheduled intervals. If incremental and total backups are scheduled on the same day, only the total backup will be performed (e.g., days between total backups=5, days between incremental backups=1; on the fifth day, both total and incremental backups are scheduled, but only the total backup is performed). The system administrator may also restore previously backed up data in a total or partial backup. If any of the data to be restored will result in overwriting existing data on the database, the system administrator will be warned prior to restoring the data, and the system administrator will be required to verify the command prior to restoring data. If an error occurs while executing a backup or restore operation, the system administrator will be notified of the problem and will be required to decide whether or not to continue, restart or abort the backup or restore procedure.

Detailed Description Text (122):
The system administrator may also perform system archival and/or retrieval of patient information records and procedure records. Archival may be scheduled as part of a specific record workflow. Additionally, the system administrator is able to create up to 16 scheduled archives, or modify/delete an existing schedule. For each archive schedule, the system administrator must provide a name for the archive schedule, the archival device, the record types to be archived, the acquiring sites, the minimum age of the records to be archived and the period and time of day for the archive to be initiated. The system administrator may also request an on-demand archival by selecting the records to be archived and specifying the archival device.

Detailed Description Text (123):
When a procedure record is archived, the system shall maintain an entry in the system database with a reference to the archival media to support retrieval of the record, sufficient data to allow for conflict detection with other records, sufficient data to support patient/procedure lists, sufficient data to support

administrative reports, a request to archive a patient folder or current demographics records shall be satisfied by archiving all unarchived procedure records for that patient. If an error occurs while attempting to archive a record, the record in the system database will remain unchanged. The system administrator is able to retrieve a selected archived record; and, if the archive media which holds the selected record is not on-line, the system administrator shall be notified that the record is unavailable and will be prompted to install the required archive media. The system administrator is able to establish a database memory utilization threshold as a whole number from 50-90%. When the percentage of database memory in use exceeds the utilization threshold, all members of the system administration group will be notified with a system message.

Detailed Description Text (131):
A report format defines one or more segments. Reports are viewed one segment at a time. If the user has the appropriate privileges, the various segments may be edited. Editing of a report segment does not affect the system database. At the user's discretion, reports may be exported, distributed, and printed. The user may schedule automatic generation and distribution of reports based on a selected format. The example shown in FIG. 24 is included generally to illustrate how finished reports are constructed from the various generated report segments. In this example, report generation is requested for the Monthly Physician Output format, which defines Report S for printing at the administrator's desk and Report T for permanent records. The report generation function generates all report segments as defined in the Monthly Physician Output format and makes them available for review. Upon user request, the function assembles the segments into two reports (as defined by the format) and distributes them to the specified locations: Permanent Records printer and the administrator's printer.

Detailed Description Text (133):
The information in tables and graphs is obtained by querying the system database to produce the desired tables or records. This section defines the basic queries available to the user, and the options available for each of those queries. The requirements in this section specify the options available to the user, but not the means of expressing those options. The defined queries are built of query elements. This section describes each of these elements. The "How many" or "Which" element distinguishes between counting the number of records which match the rest of the query (How Many) and listing the records which match the query (Which). This choice is available when defining a tabular segment. only the "How many" option is available (or meaningful) when defining graphs.

Detailed Description Text (134):
The record type list is a predefined list which allows the user to indicate which record type(s) to include in the query. The list includes "Patient Information Record" and identifies all of the installed procedure types.

Detailed Description Text (137):
The Record Throughput query is another predefined component of the administrative reports function. This query is intended to provide the system administrator with the answers to how quickly records in the system are being handled. A predefined Physician Throughput query is intended to provide the system administrator with the answers to how quickly records in the system are being handled by specific physicians. A Record Handling query is intended to provide the system administrator with information regarding the workstations being used to perform work. A Record Status query provides the system administrator with the ability to determine the status of records in the system database.

Detailed Description Text (139):
In the preferred embodiment, a Tabular Segment preferably consists of a matrix of columns and rows relating to the patient records in the system database. The row and column contents are defined by the queries. In the present invention, the user

may specify up to about 99 tabular segments for the current format. The modifiable elements of a tabular segment include the tabular segment name, tabular segment date range, tabular segment record identification or tabular segment definition. The user is preferably required to provide a unique (within the report format) name for each tabular segment. The user must indicate the time period over which data for the tabular segment will be searched. When presenting a detailed tabular display (using the "Which" query), the patient information or procedure records are included in the table. The user is able to select patient name, patient MRN, record type, record date, record time, reception date and reception time for inclusion in each Record Identification row. The user may also select the sort order for the record identifiers, choosing any of the elements selected above.

Detailed Description Text (145):
The requirements for generating each of the administrative reports segments are described herein. Report generation involves performing the necessary queries of the system database to produce the records and numbers of interest, formatting the information as specified in the administrative report format, and (if performing a scheduled report generation) distributing the report(s) as dictated by the format. The user of the present system may select any saved report format as the basis for generating the report and may generate an administrative report manually. The user may also route any manually generated administrative report. The segments are generated based on the selected report format and are generated in the orientation specified in the format (landscape or portrait). The tabular, data graph and trend graph segments include a header and footer as defined in the format for each page of the segment, and each segment is titled with the segment name. For each of the tabular segments which include a multi-page table, the table headings shall appear on each page, and each tier of a multi-tier table is indented to reflect the tier level.

Detailed Description Text (146):
The user may retrieve any saved report using the administrative reports function. For the current report, the user is able to view and edit all segments as defined in the report format and may initiate manual report distribution and notification. If serial presentation reports are requested in the report format and appropriate serial reports are available, the user is able to initiate serial presentation. The user may edit any of the free text or table entries in a report, and changes to the report will not affect the CIS system database. If the user edits automatically generated query results, totals on the generated reports will be updated. The user is also able to export the report in the ASCII format as desired. In the preferred form of the present invention, tabular segments are exported in tabular form and data will be converted to tabular form prior to export for graphical segments. The user may save the current report; if the user exits without saving the report, any changes which were made are lost. The user is also able to delete any report which has previously been saved.

Detailed Description Text (154):
The patient demographics function of the present invention provides for the viewing, printing and editing of current demographics for existing patients and the addition of new patient records. FIG. 26 shows the context diagram for the patient demographics function. The "current demographics" associated with a patient represents the most recent information on that patient. This is updated automatically when new records are received, or it may be updated manually by the user via the patient demographics function. The active current demographics are displayed on the screen for review/editing by the user. A particular patient may be represented by multiple MRN numbers if the patient has had tests run at different sites that are represented by different MRN models. Therefore, all MRNs associated with the patient along with each of the sites for which the particular MRN is valid are automatically displayed to the user. If the user has edit access to the record, the user may edit the fields of the record. Elements of the demographics which have been edited during the current session are prominently displayed (i.e., highlighted

or bolded), and each element will preferably indicate the date/time data was last entered or modified. If date of birth is undefined in the record of a patient, age will automatically be incremented by one year if more than one year has passed since the demographics have last been updated. If multiple current demographics records are selected, the user may change which of the open records is the active (viewed) record, and the user is required to save or discard changes to the active record. The user may also add a new current demographics record, and the field values of the new record will be initialized to the defaults specified. The user has the option to either save the edited current demographics record to the system database or cancel any changes that were made to the record. If the user has edited any of the MRN fields or the date of birth field, the user will be notified that the change will affect all procedure records associated with the patient. If a patient MRN field does not contain a unique identifier for the associated MRN model, the user will be informed that a unique patient MRN is required before saving, and the record will not be saved if the unique MRN is not provided. If the user accepts the changes, any changes to the MRN and date of birth fields are saved to the current demographics record and all procedure records associated with the patient. Any changes to fields other than MRN and date of birth will affect only the current demographics record. If the user cancels changes to a new current demographics record, the record is discarded. The user may export the current demographics record to a specified repository, and the ASCII format will be supported for transfer of all of the demographic records. The user may also print the current demographics record(s) currently being viewed; and, if more than one current demographics record is open, the user will be given the choice of printing the active record or all open records. The user may also print multiple copies of the selected record(s) as desired.

Detailed Description Text (175):
When a record is received on the system, the institution name and ID will be reconciled with the corresponding system site. If no association can be made, the acquiring site shall be considered "unspecified". If enabled, the location, department, physician, and technician fields will also be reconciled. If reconciliation has been enabled for the field, an attempt shall be made to match the entry to an entry in the system list. If no association can be made, the field will be assigned to "unspecified". If reconciliation has been disabled for the field, the field will be assigned to "unspecified". In the present embodiment, assigning a field to "unspecified" does not result in the loss of the original text string. If the record does not meet the qualifications of any of the defined workflows, the default workflow for that procedure will be used. The database to which a record will be stored is determined by the record's acquiring site and procedure type.

Detailed Description Text (176):
The record will also be assigned to a patient folder and stored. If the system configuration is "Standalone" or "Networked, Isolated Databases", the record will only be compared to patient folders on the assigned database. If the system configuration is "Networked, Integrated Databases", the record will be compared to all patient folders on the system. The patient demographics portion of the record is compared to the current demographics of existing patients on the database(s). The record is then assigned to a patient folder according to various patient folder assignment rules. If the record is assigned to an existing patient folder or if a new patient folder is created, the demographics of the record and/or the current demographics of the patient folder are updated. The new record contains an acquisition date and time and each field of the current demographics will have the acquisition date/time of the data stored with the field. Each field in the current demographics will then be compared to the associated field in the new record. If the field in the new record is empty or older than the current demographics field and the current demographics field is not empty, the field in the new record is updated with the value from the current demographics. If the field in the new record is non-empty and newer than the current demographics field, the current

demographics field and the associated date and time are updated with the value from the new record. If the procedure record does not already exist on the system, it is saved to the database identified for that procedure type and acquisition site. If the procedure record already exists on the system (same patient folder, procedure type, acquisition date, and acquisition time), the most recently edited record is saved to the assigned database and the older one discarded. If the system is unable to determine which record was most recently edited, both records are saved to the assigned database. If the workflow indicates that the procedure status should be updated, the record is updated to reflect the new status. If the record is unconfirmed, it is distributed as defined in the unconfirmed distribution list. If the record is confirmed, it is be distributed as defined in the confirmed distribution list. If automatic record abridgment was set up for "Immediately" the record will be abridged as specified in the appropriate procedure report. If automatic record archival was set up for "Immediately" or automatic record abridgment was set for "Upon Archival", the record is abridged as specified in the appropriate procedure report and the record is archived.

Detailed Description Text (184):
The user is allowed to view any of the reports one segment at a time. If the user is assigned the appropriate privileges, the 12-lead segment may be edited. Only one user at a time may edit a particular record. If a record is opened for edit by another user, it will only be available for viewing by other users. Editing is performed on data in the record; consequently, changes to a given data element will be reflected in all report segments that subsequently use the data element. At the user's discretion, reports may be confirmed, distributed and printed. The user may also change how the report looks by selecting a new format or changing the individual parameters of the current format. When the user is finished, the edited reports can be saved and any changes to the resting ECG record are then updated to the database.

Detailed Description Text (189):
The resting ECG reports function also allows the user to edit the header and interpretation section of the current record (serial presentation records cannot be edited). The record may then be saved, confirmed, or reconfirmed. The specific edit requirements vary based on the type of acquisition device the record was acquired on. Any changes made to a data element will be reflected in all segments in which that data element appears, and any data elements that have been edited during the current session will be displayed prominently (e.g., highlighted or bolded). The user may also save the resting ECG record to the system database identified for that record by overwriting the existing record. Additional actions, such as conflict detection, may be performed upon saving a record. If a conflict is detected, the user will be notified of the conflict along with an indication of the patient with whom the record conflicts and the nature of the conflict. The user will then have the option of completing the save (with the record marked as conflicting) or cancelling the save.

Detailed Description Text (190):
The user may also request record confirmation on unconfirmed records. In many institutions, resting ECG records may be overread by residents (without the ability to legally confirm a record) or a licensed cardiologist. If this feature is enabled in the resting ECG report setup, the user may mark a record as "overread" by entering the overreading physician's name from the physician's list, entering free text, or entering an acronym representing the physician. The user is also required to enter the name of the confirming physician by selecting from the physician's list, entering free text, or entering an acronym representing the physician. The physician list is filtered to include entries applicable only to the resting procedure. The record will then be marked as confirmed, and the user may enable or disable distribution of the confirmed record. The editing of confirmed records allow the user to save the changes to the system database without reconfirmation if the interpretation statements, diagnosis or basic statements were not changed

during editing. If any of these items were changed, the user is required to reconfirm the report in order to save the changes. The user will also be notified that the report has already been confirmed, and any changes may affect the confirmed interpretation. If the user then chooses not to reconfirm, the changes will be abandoned. If the user chooses to reconfirm, the user is required to enter the name of the confirming physician by selecting from the physician list, entering free text, or entering an acronym representing the physician. The user may also enable or disable distribution of the reconfirmed record.

Detailed Description Text (215):
The stress ECG final reports function is responsible for the generation, display, editing, printing, and routing of stress ECG reports. It is triggered by an external system client function. FIG. 36 shows the context diagram for the stress ECG final reports function. This section establishes the preferred requirements for the stress ECG final reports function. In general, the requirements specified in this section describe functions performed on stress ECG records. One or more records are selected, and this function is initiated. One report is generated for each selected record. The report is generated based on the final report format associated with the attending physician listed in the record. Each report consists of multiple segments which are appended in a specific sequence. The text document segments of a report are generated from pre-test data, event data and post-test data. The pre-test data may include data such as patient demographics and the reason for the test. The event data preferably includes information such as a ten-second ECG analysis and measurements, blood pressure data and comments. The post-test data preferably includes information such as the reason for ending test and test-maximal indications. The waveform document segments of a report are preferably generated from a combination of average-beat and ECG data. Typically, the textual information will also be included in the segment. The user may view any of the reports one segment at a time. If the user is assigned the appropriate privileges, the segments of the stress ECG may be edited. only one user at a time may edit a record; and, if a record is opened for edit by another user, it will only be available for viewing by additional users. Editing is performed on data in the record; consequently, changes to a given data element will be reflected in all report segments that subsequently use the data element. At the user's discretion, reports may be confirmed, abridged, distributed and printed. The user may change how the report looks by selecting a new format or changing the individual parameters of the current format. When the user is ready, edited reports can be saved, and any current changes to the stress ECG record are then updated to the database. In addition to the user initiated activities described above, many automatic activities such as report reception, distribution and abridgment may also occur.

Detailed Description Text (228):
With the stress ECG final reports function of the present invention, the user may edit the procedure and patient data for the active stress ECG record. The record may then be saved, confirmed or reconfirmed. Any changes made to a data element will be reflected in all segments in which that data element appears, and any data elements that have been edited during the current session will be displayed prominently (e.g., highlighted or bolded). The user is also able to edit the patient demographics for the active record and may edit stress ECG pre-test, post-test and event data. The user is also able to edit average beat measurement data and fully edit all generated narrative summary and tabular summary segments, using the data field, free text, acronym expansion and text cut, copy, move and paste text items and formatting operations of this report function. The user may also save the stress ECG record to the system database identified for that record by overwriting the existing record. The date and time of saving will be saved with the record.

Detailed Description Text (239):
The stress final report setup function is responsible for the viewing, editing

and/or printing of the final report formats, user specific review settings and record workflow parameters. If a stress ECG record is currently open or active, the user can select a new format to associate with the record or modify the format currently associated with the record. FIG. 39 shows the context diagram for the stress final report setup function. The narrative summary and tabular summary segments of the final report are user-definable segments which allow the user extensive customization capabilities to produce "signature ready" final reports. Both a narrative summary template list and a tabular summary template list are available for review by the user. In the preferred embodiment of the present invention, a default narrative summary template and a default tabular summary template are shipped with the system. The user may create new templates which may be based on an existing template. The user may also modify the existing templates (including the default templates) or delete any templates except for the default templates. If the deleted template was contained in a final report format, the appropriate default template (either narrative or tabular) will be substituted for the deleted template. The user is required to provide a name for each template that is unique to the template list and may not modify the names of the default templates. The user may use free text or data fields, such as place holders for pre-test and post-test data and also conditional text which may be embedded as if-then-else type macros or keyed off the value of specific data elements, acronym expansion, logo insertion or a stress event table having 0-9 columns with a required data field type for each column desired, and the user is required to turn the stage interval on or off when creating and/or modifying a template. If the stage interval is "off," the user is required to select a timed interval. The user will be able to choose either landscape or portrait mode for the presentation of the segment as well as set and mix fonts in the segment and perform standard word processing operations (such as cut and paste) when creating and/or modifying a template.

Detailed Description Text (242):
The user may even select a tabular summary template from a list of summary templates. If the current lead format is standard or Cabrera 12-lead, the user may select a maximum deviation report in a 6.times.2 or 3.times.4 format. If the current lead format is a three lead format, a 3.times.1 format will be automatically be used. The user may then select stage only or stage+time wherein the time interval shall be specified in 10 second increments, ranging from 00:10 to 99:50 as reporting intervals for the Average Beat Summary. If the current lead format is standard or Cabrera 12-lead, the user may select the number of channels (three or six) and may select a lead for each channel from current lead format or none. The user will also be able to determine which types of averages will be included in the summary. These averages preferably include current averages only or current and resting averages. The trend graphs segment may be defined by using up to 36 graphs, and the user is required to select one data field type from the in-test tabular test data for the Y axis of each graph (the X axis is always time). If the data field type "blood pressure" is selected, both diastolic and systolic pressures are plotted. The user may also determine the position on the page in which selected graphs will be presented. Typically, users will also want to include representative recordings with their final report. The selections described below will act as filters to reduce the number of recordings included with reports by default. The user may override any of these selections during the report review process. The user is able to limit which ECG recording types will be included by enabling or disabling inclusion of the 12-lead, Exercise Set, Average Beats, Write Screen, Rhythm, Ectopic or Write Hold ECG recordings segments. The user may also limit which major recording events will be included by enabling or disabling inclusion of Resting (recordings corresponding to the resting averages), Peak (recordings occurring at time zero of recovery phase, or at the end of exercise if no recovery) or Final (the last recordings of recovery phase) major recording events. The user may also limit which mode of recordings will be included by enabling/disabling inclusion for programmed or manual modes. The user may also select which segments are present and the order of the segments in the printed

final report. All segments are always present for review.

Detailed Description Text (251):
An important workstation framework concept is the abstract concept of records composed of fields and stored on data repositories (i.e., databases). Although portions of these concepts, such as fields, are substantially implemented within the workstation framework modules, the workstation framework primarily provides building blocks (classes) that can be used by the workstation products to implement these concepts. For instance, the workstation framework modules do not implement any records but do provide abstract base classes that can (and must) be used by workstation products to implement records.

Detailed Description Text (254):
The following abstract workstation framework concept of records, composed of fields and stored on data repositories (i.e., databases), presents the abstract framework record concepts from multiple viewpoints. The Applet views a record object as being simply a collection of field objects. The record views itself as an accessor object capable of accessing field objects on some data repository. The Database Accessor views records as a collection of recordset objects to some database. The recordset views fields as a mapping of the database contents.

Detailed Description Text (256):
As shown in FIG. 42, each record object contains an accessor object. In one form of the present invention, the only supported accessor type is the Database Accessor. From a record's viewpoint, the accessor object consists of multiple fields which are always available. It appears to the record object that the fields are retrieved directly from the accessor object. An accessor can be thought of as an abstraction of the persistent storage of a record. When a record object receives a request for a field object, it asks its accessor object to provide it. The accessor object appears to construct and return the requested field object if it is available. When a record object is told to save itself to persistent storage, it requests its accessor object to save each field object to persistent storage. These relationships are shown in FIG. 42 where the arrows show calls made, in the direction of the calls.

Detailed Description Text (257):
As shown in FIG. 43, an example of an accessor type is the Database Accessor. Each Database Accessor object contains one or more recordset proxy objects. Each recordset proxy object exposes all the methods of a recordset object but constructs the associated recordset object only if access is requested to a field contained in that recordset object. Thus, the storage required for the data contained within a recordset object is allocated only if an Applet actually uses that data. This technique of constructing recordset objects only as fields are requested from them is called lazy construction. It has the potential of making significant reductions to database traffic and to the workstation's memory requirements. From the Database Accessor object's viewpoint, each recordset proxy object consists of multiple field objects which exist and are available upon the construction of the recordset proxy object. When a Database Accessor object receives a request for a field object, it asks each of its recordset proxy objects for this field until either the field object is constructed and returned by a recordset proxy object or all recordset proxy objects have been asked for the field object. Requests of the Database Accessor object to save a field back to persistent storage are forwarded to each recordset proxy object until either a request made to a recordset proxy object succeeds or all recordset proxy objects have rejected the request. When a Database Accessor object is asked to save itself to the database, it asks each recordset proxy object in turn to save itself to the database. Any recordset proxy object that does not yet contain a recordset object ignores these requests and returns successfully. These relationships are shown in FIG. 43 where the arrows show calls made, in the direction of the calls.

Detailed Description Text (258):
As shown in FIG. 44, each recordset object contains multiple member data items, including one distinct member data item for each and every field which a recordset object can process. From a recordset object's viewpoint, these member data items are the actual database data elements representing row-column intersections within a database table. Each member data item represents a unique column. All member data items are always in the same row of the database table. Any changes made to the individual member data items are automatically made back to the database, either immediately or when the database is asked to save the recordset object. When a recordset object is asked to construct a field, it calls the field factory within the FieldFramework module to construct an actual field object. It then initializes that field object with the appropriate member data item. Requests to save a field to the database cause the recordset object to retrieve the field object value and save it into the appropriate member data item. These relationships are shown in FIG. 44 where the wide white arrows show calls made in the direction of the calls, and the black arrows show data flow.

Detailed Description Text (259):
In the preferred form of the present invention, the primary persistent storage repository for "records" may either be single sequel database servers or multiple sequel database servers depending on the configuration of the CIS. The preferred primary persistent storage repository for centralized configuration is a WINDOWS NT Domain Server Registry. Examples of centralized configuration information include definitions of users and groups, privileges assigned to individual users and groups, and system configuration settings such as name format. Additionally, directions on a WINDOWS NT file server may be defined to include ordinary disk files such as archived records, data transfer files, facsimiles or scanned images and saved administrative reports. FIG. 45 illustrates an example of the workstation framework for the client/server processes. In this example, multiple client workstations and a server workstation are connected to a Token Ring configuration, and a remote client workstation is remotely connected to the server workstation.

Detailed Description Text (260):
FIG. 46 illustrates the processes of the client workstation and the server workstation. As shown, the client workstation includes Applets for the Client Shell processes. The Service Shell processes and the server workstation include sequel server processes as well as Applets for the Service Shell processes. FIG. 47 is illustrative of the database communication processes of the client and server of the preferred embodiment. FIG. 48 is illustrative of the inter-shell communication of the client and server processes.

Detailed Description Text (261):
The workstation framework preferably includes various modules therein. As shown in FIG. 49, the workstation framework includes software modules for the Framework Shell, Framework Applet, dynamically loadable Framework Applets and dynamically loadable CIS Applets. The Framework Shell preferably includes the Client Shell or Service Shell and an Applet interface layer consisting of the Applet interface and Applet loader modules. The Framework Applet further consists of the services layer, the rendering layer, the manipulation layer and an access layer. The services layer preferably includes the Event Logger Applet and Access Services Applet modules. The rendering layer preferably includes record presentation Applet and Applet Widgets Applet modules. The manipulation layer preferably includes Field Framework Applet and Record Framework Applet modules. The access layer preferably includes a Database Access Applet module. The dynamically loadable Framework Applet preferably includes the Admin Reports Applet, Patient Demographics Applet, Record List Applet and transfer SCP Applet modules. The dynamically loadable CIS Applet includes the Rest Applet and Stress Applet modules. FIG. 50 illustrates the modules of the workstation framework software which preferably are run within the Client Shell environment. FIG. 51 illustrates the modules of the workstation framework software which are preferably run within the Service Shell environment.

Detailed Description Text (271):
The Manipulation Layer preferably consists of the FieldFramework and RecordFramework modules. The FieldFramework Applet provides services which encapsulate data-specific knowledge about a specific data element and which format that data element for display. Each data element on the database can be represented as a field object which knows how to format and manipulate the data element contained within it. The FieldFramework may also be enhanced to support compound fields containing multiple, closely related data elements (such as a name or address) and to support array fields containing arrays of like data elements. The FieldFramework services are preferably used by all Applets which need information representing database data elements. This includes the RecordFramework and DatabaseAccess Applets as well as the dynamically-loadable Applets. The FieldFramework Applet is preferably a statically-loaded Applet DLL, and the other Applets are typically statically linked to the FieldFramework Applet. The FieldFramework services are used by the RecordList, PatientDemographics, RecordFramework and the DatabaseAccess modules.

Detailed Description Text (273):
The Access Layer preferably provides access to various persistent storage media. It implements the transfer of data between a record object and a specific persistent storage medium. In the present embodiment, the Access Layer consists of the DatabaseAccess Applet. In the preferred embodiment, the DatabaseAccess Applet provides access to the CIS database which is implemented using the Microsoft SQL Server. The MFC database classes, which internally use ODBC, are used for communication with the SQL Server. The DatabaseAccess Applet is preferably a statically-loaded Applet DLL, and the other Applets are typically statically linked to DatabaseAccess. The DatabaseAccess services are used by the RecordList, RecordFramework and PatientDemographics modules.

Detailed Description Text (274):
In the preferred embodiment, all Dynamically-Loadable Applets are dynamically loaded by AppletLoader, and Dynamically-Loadable Applets are found only if they reside in the same directory on disk as shell. It is anticipated that the mechanism for identifying Dynamically-Loadable Applets may be extended to allow for the searching of a different directory or additional directories or by limiting the loading to a subset of the Dynamically-Loadable Applets available. In the present embodiment, there are preferably three Dynamically-Loadable Applets. The AdminReports Applet provides administrative reporting and is a Dynamically-Loadable Applet DLL. The RecordList Applet is a Dynamically-Loadable Applet DLL which provides windows containing lists of records on the database. The PatientDemographics Applet is a Dynamically-Loadable Applet DLL which provides patient demographics related records. The records in these lists can be selected for processing, and operations can be performed on these selected records. An explorer-like window is provided which displays available Record Filters. This window allows the selection of a Record Filter and displays the results of a database query represented by the selected Record Filter.

Detailed Description Text (275):
In the preferred form of the present embodiment, two Record Filters are provided. One Record Filter displays all procedure records on the database, and the other Record Filter displays all patient records on the database. Submenus are provided on the ClientShell Lists menu to activate the explorer-like window or to directly activate a selected Record Filter within an independent child window. In the present embodiment, patient records and current patient demographics records may be processed. Selected patient records may be opened, creating a patient folder window for each selected patient record. Selected current patient demographics records within patient folders can be selected for editing or viewing by PatientDemographics. Doing so causes RecordList to request the Record Factory to construct the selected current patient demographic record object. The Record

Factory in turn calls the registered Record Builder provided by the PatientDemographics Applet to construct the actual record object. The record object, once constructed, is asked by RecordList to edit or view itself, causing the edit or view request to be processed by the PatientDemographics Applet. The PatientDemographics Applet is a Dynamically-Loadable Applet DLL which provides a Record Builder, registered with the Record Factory (within RecordFramework), that is capable of constructing record objects representing current patient demographics records on the database. The PatientDemographics Applet builds upon RecordPresentation and AppletWidgets to provide editing and viewing of patient demographics fields contained within current patient demographics records, and changes made during edit are saved to the database.

Detailed Description Text (368):
FIG. 85 illustrates an overview of the relationships between the classes defined in the Database Access module and the relationships between classes within the Database Access module and classes within other workstation framework modules. Those classes whose inheritance is not explicitly shown are derived from CObject. The Database Access module provides an implementation of database recordsets which does not throw exceptions, but instead returns error codes to the caller. The class CdaxCRecordsetWrapper provides the functionality of the MFC class CRecordset, but catches all of the exceptions thrown by the CRecordset class and "translates" the exceptions to error codes. The class CdaxCRecordsetWrapper is implemented by containment of a CRecordset-derived class. The subset of the CRecordset-like methods which are provided by the Wrapper class simply call the corresponding CRecordset methods, with a "try/catch" wrapper around those methods which throw exceptions. It may seem natural to simply derive CdaxCRecordsetWrapper from CRecordset, but it does not work because CRecordset catches many of its own exceptions; and in some cases it throws a new exception, and in other cases it does not. If a derived class caught exceptions for some of the methods before CRecordset had a chance to catch them, then the behavior of CRecordset may be changed. By containing a CRecordset class, the exceptions are caught only at the outer layer. Additionally, changing the interface from that of exceptions to errors suggests changes to the public interface of the class. For example, CRecordset's Edit method is a void function which may throw exceptions. The CdaxCRecordsetWrapper changes this method to return BOOL, indicating success or failure; if the method fails, the user may query the class to see the nature of the error, which is translated from the exception thrown by the contained CRecordset. The inheritance is only advisable if the derived class maintains the interface of the base class. As shown, the Class CdaxCRecordsetWrapper does not actually contain a CRecordset class, but rather a class derived from CRecordset and CdaxCRecordset. This is preferred because CRecordset defines some pure virtual methods. The class CdaxCRecordset provides an implementation for the pure virtual methods of CRecordset. These CdaxCRecordset methods invoke the corresponding CdaxCRecordsetWrapper methods via a pointer to the CdaxCRecordsetWrapper object. The class CdaxRecordset derives from CdaxCRecordsetWrapper and adds the ability to store data to/from CffField objects. The CdaxRecordset is an abstract class, and it is intended as the base class for the concrete recordset classes used by framework-based applications. These derived concrete classes may be generated by the MFC Class Wizard.

Detailed Description Text (372):
The following discussions contain interface descriptions for the classes defined in the Database Access module. The Database Access module provides abstractions for getting information to and from a relational database via an ODBC driver. It incorporates and expands upon the features of the MFC classes CDatabase and CRecordset. The classes CdaxDatabase and CdaxCRecordsetWrapper catch exceptions thrown by MFC classes CDatabase and CRecordset. These exceptions are converted to error codes. These classes include the global definition "daxErrorCode" which is a global enumeration which defines the error codes that can be generated by methods within the Database Access module.

Detailed Description Text (373):
The class CdaxApplet is from the Database Access module and preferably provides the Applet interface for the Database Access module. On Applet initialization, the database connection for the "default" database is established. The default database is the database to which new records or records received via Data Transfer are stored. This Applet provides an interface for other Applets to request information about the available databases. The class CdaxAccessorKey is declared a friend to this class; the CdaxAccesssorKey class is used to get a connection to one of the available databases. The class CdaxApplet includes various public methods such as "CdaxApplet," ".about.CdaxApplet," "InitApplet", "ExitApplet," "GetAppletName," "GetApplettitle," "GetNumDatabases," "GetDatabaseTitle," "GetDatabase" and "releaseDatabase."

Detailed Description Text (374):
The class CdaxDatabaseMgr is from the Database Access module and preferably manages multiple connections to a single database source; each thread which needs access to a particular data source gets its own connection to the data source. This class is defined locally within CdaxApplet and is only usable by CdaxApplet. The CdaxApplet class creates one of these objects for each of its data sources. The class CdaxDatabaseMgr includes various public methods such as "CdaxDatabaseMgr," ".about.CdaxDatabaseMgr," "GetDatabase" and "ReleaseDatabase."

Detailed Description Text (375):
The class CdaxDatabase is from the Database Access module and preferably provides database connection support. This class is derived from the MFC class, CDatabase, to which it additionally catches exceptions which can be thrown by CDatabase's "open" method, by overriding the "open" method and wrapping a call to the base class method with try/catch. This class also performs direct ODBC calls during initialization to allow cursor preservation during transaction processing and sets up the base class, CDatabase, to allow transaction processing and which overcomes a shortcoming in the CDatabase/ODBC interface. This class also adds "time of connection" functionality and adds a "use count" attribute which monitors how many CdaxAccessorKey objects are connected to a CdaxDatabase object. When the user count goes to zero, the object may be destroyed. Additionally, this class supports a unique connection for each thread which requires database access. Creation of objects of this class is intended to be done only by the Database Access Applet. The class CdaxDatabase includes various public methods such as "CdaxDatabase," ".about.CdaxDatabase," "SetThreadID," "Open," "IncrementUseCount," "DecrementUseCount," "GetLastError," "GetLastException" and "GetConnectionTime."

Detailed Description Text (376):
The class CdaxAccessor is from the Database Access module and preferably is the implementation of the CrfAccessor base class which interfaces via the Database Access module to the ODBCSQL database. The class CdaxAccessor preferably includes various public methods such as "CdaxAccessor," ".about.CdaxAccessor," "GetField," "RefreshField," "StoreField," "IsReadOnly," "Lock," "Unlock," "QueryLock" and "Save."

Detailed Description Text (377):
The class CdaxAccessorKey is from the Database Access module and preferably is derived from the CrfAccessorKey. It defines a database connection through the Database Access (DAX) layer and the primary key which identifies a particular record on that database. This class preferably includes various public methods such as "CdaxAccessorKey," ".about.CdaxAccessorKey," "SetThread," "operator=," "operator==," "operator!=," "GetKeyString," "GetDatabase," "IsPrimaryKeyValid," "GetPrimaryKey," "SetPrimaryKey" and "InvalidatePrimaryKey."

Detailed Description Text (378):
The class CdaxRecordsetKey is from the Database Access module, and this class preferably provides a means of identifying the particular row(s) of a database

which correspond to a particular record. This class specifies the database key for the Recordset. This class preferably includes various public methods such as "CdaxRecordsetKey," ".about.CdaxRecordsetKey," "operator==," "operator!=," "IDBKeyValid," "GetDBKey," "SetDBKey" and "InvalidateDBKey."

Detailed Description Text (379):
The class CdaxCRecordset is from the Database Access module and preferably serves as an intermediary between classes CdaxCRecordsetWrapper and CRecordset. This class provides a concrete CRecordset-derived class which may be contained by class CdaxCRecordsetWrapper. It modifies the CRecordset by requiring a valid pointer to a CDatabase object to be provided for class instantiation. The CRecordset constructor provides a default NULL value for CDatabase pointer; this class provides no default and does not allow a Null value. The CRecordset is also modified to require that a pointer to a CdaxCRecordsetWrapper object be provided for class instantiation. This is necessary so that the CdaxCRecordset class can invoke methods of its creating class. The CRecordset class is also modified to provide an implementation of the pure virtual methods of CRecordset class, GetDefaultSQL() and DoFieldExchanger(). This creates a concrete class which can provide the CRecordset functionality. These implementations call corresponding methods of CdaxCRecordsetWrapper class. The CdaxCRecordset class also preferably includes various public methods such as "CdaxCRecordset," ".about.CdaxCRecordset," "SetOwner," "GetDefaultSQL," "DoItFieldExchange," "SetNumFields," "SetNumParameters," "SetFilterString," "SetSortString" and "SetDefaultType."

Detailed Description Text (380):
The class CdaxCRecordsetWrapper is from the Database Access module and preferably provides a CRecordset-like interface and functionality, but catches all of the exceptions thrown by CRecordset and converts them into error codes. It also exposes a subset of the CRecordset interface. The CdaxCRecordsetWrapper class also preferably includes various public methods such as "CdaxCRecordsetWrapper," ".about.CdaxCRecordsetWrapper," "GetLastError," "GetLastException," "GetDefaultSQL," "DoFieldExchange," "CRecordset methods," "CRecordset methods with error status," "CRecordset methods with return status" and "CRecordset methods with return status added." The CdaxCRecordset class also preferably includes various protected methods such as "SetNumFields," "SetNumParams," "SetFilterString," "SetSortString" and "SetDefaultType."

Detailed Description Text (381):
The class CdaxRecordset is from the Database Access module and preferably provides mechanisms and support for transferring recordset data to or from CffField objects. It is an abstract class from which all the concrete workstation recordset classes are derived. Each derived class provides to this base class an array of the Field IDs which are represented in the derived recordset and a handler for each of the Fields; the handler is used to transfer data between the derived recordset and a Field object. The CdaxRecordset class preferably includes various public definitions such as "enum RECORDSET.sub.-- TYPE" and "FIELD.sub.-- NOT.sub.-- FOUND." The CdaxRecordset class also preferably includes various protected definitions such as "enum DATA.sub.-- DIRECTION" and "struct FIELD.sub.-- HANDLER." The CdaxRecordset class also preferably includes various public methods such as ".about.CdaxRecordset," "GetField," "RefreshField" and "StoreField." The CdaxRecordset class also further preferably includes various protected methods such as "CdaxRecordset," "BuildFilterString," "BuildSortString," "FindField" and "AtomicFieldHandler."

Detailed Description Text (382):
The class CdaxRecordsetProxy is from the Database Access module and is preferably an abstract base class which defines the interface for the CdaxProxy template classes. The CdaxRecordsetProxy class preferably includes various public methods such as "CdaxRecordsetProxy," ".about.CdaxRecordsetProxy," "IsRecordsetAvailable," "HasField," "GetField," "RefreshField," "StoreFiled" and "Save."

Detailed Description Text (383):
The class CdaxProxy is from the Database Access module and is preferably a template class which serves as a "smart" interface to a CdaxRecordset-derived object. It takes, as a template parameter, a class derived from CdaxRecordset; operations which can be performed on a CdaxRecordset object are passed through the Proxy object, but the Proxy first checks that the Recordset object is instantiated and instantiates it if necessary. Thus, the owner of the Proxy object can deal with the Recordset as if the recordset were always available, and the Proxy makes up the difference. The CdaxProxy class preferably includes various public methods such as "CdaxProxy," ".about.CdaxProxy," "IsRecordsetAvailable," "HasField," "GetField," "RefreshField," "StoreFiled," "Save," "operator->" and "GetRecordset."

Detailed Description Text (384):
The class CdaxCompositeFieldHelper is from the Database Access module and preferably is a class which supports the transferring of data between a Recordset and a composite Field object. This class provides an overloaded "SetSubField" method for each of the atomic data types. The CdaxCompositeFieldHelper class preferably includes various public methods such as "CdaxCompositeFieldHelper," ".about.CdaxCompositeFieldHelper," "GetSubField" and SetSubField."

Detailed Description Text (385):
The class CdaxRecordsetList is from the Database Access module and preferably provides support for "secondary" database tables which are referenced in other tables. For example, a CdaxRecordsetList-derived class might encapsulate the PERSONNEL table of the database. This derived class would then provide a static method for obtaining the personnel data given a PersonnelKey. These RecrodsetList classes may be available as "Fully cached" where the encapsulated table is read from the database at system initialization and cached; "As used," where as elements are requested from the encapsulated table, they are cached; "MRU cache," where elements are cached in a "Most-Recently-Used" fashion; or "Uncached," where each element is retrieved from the encapsulated table upon request.

Detailed Description Text (386):
The derived CdaxRecordset classes are from the Database Access module, and the class CdaxRecordset is preferably an abstract class from which all concrete recordset classes are derived. This section outlines the members of a hypothetical class derived from CdaxRecordset and CmyRecordset. This class preferably includes various public methods such as "CmyRecordset," ".about.CdaxRecordset" and "HasField."

Detailed Description Text (401):
The field framework module of the workstation client software portion of the workstation products framework product is described in further detail below. This section also describes the CffField hierarchy defined in the field framework module as well as the relationships between classes within the field framework module and classes within other workstation framework modules. The Field Framework is preferably an Applet DLL which encapsulates data-specific knowledge and serves as the data formatter. Each data element on the database can be represented as a field object with that object knowing how to format and manipulate the data element contained within it. As shown in FIG. 92, the CffField provides the primary interface between Field Framework and the other modules of the workstation products framework. The CffField derived classes are used to encapsulate the specific data representation with the access to this information done through virtual CffField class methods. This allows users of fields to use them as if they were all of the same class instead of multiple derived classes.

Detailed Description Text (402):
The field framework, and specifically each field object, is responsible for any data formatting that may be necessary in order to properly display, print or store

the data contained within the field. The formatting information for each Field Id is maintained in tables. This information may be loaded from the database on initialization. The information for each Field ID includes the minimum allowable value for this field, the maximum allowable value for this field, the default value for this field, the Field Type and the list of acceptable values for Enum type fields. The specific information retrieved into these tables may vary by Field Type and may expand to include more than what is described herein. The desire to store this information on the database is driven by the desire not to duplicate the effort of maintaining and deriving this information. It is preferably designed once and then maintained on the database. Also, the loading of these items may be made to include all fields which are currently known to the database. Since other Applets may add fields (either derived classes or new Field IDs or both), there may be new fields in the future which the current field framework is unaware of. If the framework were responsible for maintaining the table information, the framework would need to change for every new field incorporated into the system. With the database storing this information, the field need only retrieve this information. Also, at initialization, the field framework Applet begins an array of function pointers to field builders which are to be added to by other Applets. The field builders are methods which construct specific fields based on a Field Id. Any Applet which wishes to define a new type of field, which will be derived from CffField, must also supply a method by which that field can be constructed. This removes the burden from the field framework for having to know about all types of fields. By allowing other Applets to define their own fields, the other Applets will be able to extend the field framework without directly affecting the basic structure of the field framework as shown in FIG. 93.

Detailed Description Text (403):
One goal for the field framework design is the encapsulation of the data representation. Data representation encapsulation is preferred so that changes to a data element's data representation do not ripple throughout the software code. The data representation for each field is influenced initially by the database with the understanding that the database representation may change in the future. Since a user of a field is responsible for initializing the internal data of the field, a method which sets the value in the field is necessary. This method preferably includes a typed parameter. Also the contents of the field are preferably retrievable thereby necessitating a method for getting a typed value from the field. If these methods are public, the main goal of encapsulation of the data representation has been compromised as any owner of fields may retrieve a typed copy of the data stored in the field. However, disallowing owners of fields to manipulate the value within the field is undesirable as well. Therefore, a compromise solution which can partially fulfill both needs must be achieved by the Field Framework design.

Detailed Description Text (405):
Composite fields are groupings of other fields. They differ from atomic fields in that they contain other fields. Fields defined in the Field Framework are predominately atomic and map to the atomic data representations of the database. For example, on the database there is a signed 4-byte integer type so a CffIntField with a signed, 4-byte internal representation has been defined. However, future Applets may wish to define new field classes as composite. The field framework must be designed to allow this expansion without modifying the framework for each new occurrence of a composite type.

Detailed Description Text (407):
Each of the name part fields can handle formatting and checking itself while the composite field is able to format the whole name. For example, using the values defined above, a call to the GetFormattedString method of the CffNameField Object would return "Linda J. Shoemaker" or "Shoemaker, Linda J." depending on the defined format. The defined format may be determined by a system API call or by a field from the database. A call to the GetFormattedString method of the CffCStringField

Object for the FID.sub.-- PAT.sub.-- MIDDLE.sub.-- NAME might return "J.". Although a rule about placing a period after a single letter middle name may be encapsulated within the field object since the field for the middle name in this example is a generic type of field, there is no place for the rule to be encapsulated. To encapsulate the rule, FID.sub.-- PAT.sub.-- MIDDLE.sub.-- NAME would have to use a field class which is derived from CffCStringField and adds the desired behavior. This last call necessitates that the user of a CffNameField must be able to access the objects within it. Methods within CffField and overridden in derived classes allow the user to retrieve from a given field object a list of the ids for the fields which this object contains. This is shown as m.sub.-- fidlist in FIG. 94B. The user will also be able to retrieve any sub-field object of a given field using the sub-field id. These two methods allow the user to traverse a composite field to any depth. In this name example, the owner of the CffNameField Object will be able to get the m.sub.-- fidlist and then get each of the field objects using this list. Once the desired field is retrieved, the owner may invoke methods which manipulate the underlying data element.

Detailed Description Text (411):
It is anticipated that the CffArrayField class will be suitable for all types of arrays and that it will not need to be subclassed. However, there is nothing in the design of the present embodiment to prevent a subclass of the CffArrayField class. There will be a member attribute which specifies the Field Id of the elements which can be added to the CffArrayField object. The Field Id of the elements is dependent on the Field Id of the CffArrayField object. So, for example, the FID.sub.-- PAT.sub.-- MED.sub.-- ARRAY would be of type CffArrayField, and the only elements which could be added to the object would be FID.sub.-- PAT.sub.-- MEDS. All of this default information is stored on the database.

Detailed Description Text (413):
The field framework provides a number of enum definitions for use with fields. These enums exist on a global scale due to their wide use and the cumbersome task of always scoping them. The enums for Field ID (FID) are used to uniquely identify each element on the database. The FIDs are heavily used throughout the client software to refer to specific data elements. The enums for FLD.sub.-- TYPE identify the base storage type which will be used for each defined FID. As the design of the preferred embodiment expands and grows to include further modules, it is anticipated that the number of FLD.sub.-- TYPEs will grow. The enums for FLD.sub.-- ERR identify the errors that may occur within the Field Framework. The enums for FLD.sub.-- FLAG identify how the field should treat a particular situation or value.

Detailed Description Text (417):
The class CffNameField is from the Field Framework module and preferably is a composite field which groups the components of the any name. The formatting of the name is also handled with this field. The name of the patient and the name of the physician as well as the names of other entities are identical in the components and the formatting of those components. This field will support any of these by using the database to indicate which FIDs correspond to which collection. In other words, the database will maintain a list of the FIDs which go with FID.sub.-- PAT.sub.-- NAME and a list of the FIDs that go with FID.sub.-- PHY.sub.-- NAME. The FIDs for the components are different because they represent unique quantities on the database. However, for formatting purposes, those components can be treated identically. However, because the FIDs are actually different from one name to another, the order in which they appear in the list of FIDs (on the database and in the field object) is critical. If the order is wrong, the formatting will be wrong. The class CffNameField preferably includes various public methods such as "CffNameField," ".about.CffNameField," "GetFormattedString," "GetSubField" and "GetFIDList."

Detailed Description Text (436):

The class CpdRecordset is from the Patient Demographics module and preferably provides the database interface necessary to retrieve and store patient records from and to the database. This class preferably includes public methods such as "CpdRecordset," ".about.CpdRecordset," "HasField," "GetDefaultSQL" and "DoFieldExchange."

Detailed Description Text (437):
The class CpdPatientFrmView is from the Patient Demographics database and preferably provides a view object in which to display fields within the Patient Demographics Record. It is anticipated that in other embodiments of the present invention, this class may be removed and replaced with elements from the Record Presentation Applet. The content of the CpdPatientFrmView class is governed by the UI. The CpdDisplayFrame class handles all of the messages so no message handlers are defined in this view. The class CpdPatientFrmView preferably includes various public methods such as "CpdPatientFrmView" and ".about.CpdPatientFrmView."

Detailed Description Text (442):
From the viewpoint of the DaxAccessor, the database access layer is a collection of Recordsets each of which contains Fields. When the owner of a Record (whose Accessor is Dax) requests a Field, the Record queries its associated DaxAccessor object, which in turn queries each of its Recordsets until one of those Recordsets responds with a valid Field object to satisfy the request. When the Accessor returns a Field object, the Record places that Field and the Field ID in a local map. This map is then used to access the Field on subsequent requests with the same Field ID. It is important to note that in the preferred embodiment of the present invention, the DaxAccessor object does not actually contain Recordsets, but rather it contains objects called RecordsetProxies. Each Proxy is a stand-in for the Recordset because the Recordset object itself is not instantiated, and the database query for that Recordset is not performed until one of the Fields from that Recordset is requested. This is the "lazy construction" mechanism referred to above that greatly reduces the number of database hits and the size of records on the client. In FIG. 101, it is presumed that Fields with identifiers 8, 14 and 123 have been requested of a Record whose Accessor is Dax. This has caused Recordsets 2 and 3 to be instantiated and populated from the database server to satisfy those requests. Pointers to the Field objects returned from the Recordsets have been placed in the Record's Field map. Since no Fields have been requested from recordset.sub.-- 1, it is not instantiated. To continue the example shown in FIG. 101, the following actions will occur in response to requests for different Fields:

Detailed Description Text (443):
As shown in FIG. 102, the Record List Applet presents a list of records to the user, allowing the user to select one or more entries from the list and perform some operation(s) on them. While in the Record List, the entries are not CrfRecord objects; they simply reflect the result of a database query. To perform an operation, the selected Record List entries must be instantiated as CrfRecord objects; this is performed through the services of the RecordFactory. These objects are then placed in a CrfRecordContainer object, and the desired operation is performed on each of the record objects within the container by invoking the "DoOperation" method of the container.

Detailed Description Text (450):
The class CrfAccessor is from the Record Framework module and is preferably the base class for RecordFramework Accessor classes. An Accessor class makes transparent the implementation of the persistence for a CrfRecord-derived object. It is intended that this class be derived for different persistent storage types such as databases, files, SCP, etc.; and, if necessary, derived from there to support the derived Record objects. The CrfAccessor class preferably includes public methods such as "CrfAccessor," ".about.CrfAccessor," "GetAccessorKey," "GetField," "RefreshField," "StoreField," "IsReadOnly," "Lock," "Unlock,"

"QueryLock", "IsLocked," "PrepareToSave" and "Save."

Detailed Description Text (456):
The class CrfAssocProc is from the Record Framework module and is preferably a helper class used as an aid in creating a CrfRecordContainer which is populated with Procedure Records which are related to some other record (such as Patient or Procedure). All of the Assoc records selected by this class belong to the same patient. The class constructors essentially set up database queries with which to populate a container. The records are stored within the container with the oldest record first and the most recent record last; this order allows the "Next" and "Prev" methods to have the correct temporal meaning. This class preferably includes public definitions such as "ALL.sub.-- RECORDS," "THIS.sub.-- PROC.sub.-- TYPE" and "ALL.sub.-- PROC.sub.-- TYPES" as well as public methods such as "CrfAssocProc," ".about.CrfAssocProc" and "GetRecordContainer." This class also preferably includes private methods such as "CrfAssocProc" and "operator=."

Detailed Description Text (457):
The class CrfProcRecord is from the Record Framework module and preferably provides the basic functionality for procedure records. This is an abstract base class to be derived for actual procedures. It is derived from the CrfRecord class and adds two recordsets. One of the recordsets is used for basic patient information, and the other recordset is used for basic procedure information. This class preferably includes public methods such as "DoPublish," "DoPreSubscribe" and "DoPostSubscribe."

Detailed Description Text (458):
This section discusses the relationships between the classes defined in the Record List module as well as the relationships between classes within the Record List module and classes within other Cardiology Information System modules. The crux of what Record Lists do is provide the user with a list of records from which to choose according to a selected hierarchy. The information contained in the list should be enough for the user to uniquely identify the record and should be ordered (sorted) in such a way so as to make the list usable. These two items, information and information order, make up what is referred to herein as a filter.

Detailed Description Text (460):
FIG. 104B shows the relationships between the various view classes for the record list module. All views are derived from the MFC class CView. The CrlListView presently contains a CrlListCtrl which manages the listing of all the records returned from the database but also may include other data sources as desired. The CrlExpListView, which is derived from CrlListView, has an additional control, a CEdit Control. The CrlBtnView contains a CawBtnBar which manages the buttons for user selection. The specific buttons displayed depend on the current set of records in the CrlListCtrl. The CrlTreeView contains a CTreeCtrl. The specific views instantiated are determined by the frame.

Detailed Description Text (462):
As shown in FIG. 104D, Populators may be used to fill the CrlListCtrl object in a CrlListView or a CrlExpListView. A populator is used to hide the source of the list items. The list may be filled from the tree control or from a recordset or other data sources as may be desired. The populator is preferably contained within the CrlListCtrl and may be swapped out for another populator when the user actions suggest a different data source. The CrlFromTreePop class is able to provide the needed list items from the CTreeCtrl class within the CrlTreeView class. This is preferably only used in the Explorer context. The CrlPatListPop class provides the CdaxRecordset derived classes necessary to get the needed information from the database for a list of patients. The CrlPatFolderPop class provides the CdaxRecordset derived classes necessary to get the needed information from the database for a list of all procedure records and the patient demographics record for a given patient which may be selected from a list of patients. The

CrlProcListPop class provides the CdaxRecordset derived classes necessary to get the needed information from the database for a list of procedure records. The populators provide the header information, if any is needed, for the CrlListCtrl class and provide the CrlListCtrl class with formatted strings to use in populating the control. In opening the recordsets, the populators use information from a filter object to determine the sort criteria for the recordset.

Detailed Description Text (464):
During Record List Applet initialization, several tasks must be accomplished. First, a CrlManager object must be instantiated. Then all the CrlFilter objects must be created and initialized from values which will preferably be on the database, although they may also be hard-coded. Once the filter objects are created, the menu and submenu items can be created. There is only one menu item, and the name for that menu item will be a string resource for "Lists." The submenus will be objects of a CapSubmenu derived class. The derivation from CapSubmenu is preferably necessary to add the command handler routine. The derived class, CrlSubmenu, will also contain a CrlManager pointer because all commands will be passed to the CrlManager object for further handling. FIG. 105 illustrates the classes which participate in the initialization process.

Detailed Description Text (467):
The Record List Applet uses field objects as tools to accomplish the conversion of a data element from the data representation on the database to the CString object needed to build the list. FIG. 106C shows some of the internal implementation of the populator classes. In this figure, a filter object is pointed to by the m.sub.-- pFilter attribute of the populator. The CffField objects within an array in populator are in the order specified by the column order of the filter object. The field objects are created and populated through the contained CrlPatRecordset class.

Detailed Description Text (470):
The class CrlApplet is from the Record List module and preferably initialization of items from the database and initialization of new menus and submenus needed for this Applet. The class CrlApplet preferably includes public methods such as "CrlApplet," ".about.CrlApplet," "InitApplet," "ExitApplet," "GetAppletName," "GetAppletTitle" and "GetClientConfig."

Detailed Description Text (473):
The class CrlFilter is from the Record List module and preferably provides encapsulation of the information stored on the database (referred to as a filter) about what information is to be retrieved from the database and in what order it is to be displayed (both row and column). A CrlFilter object is only used in certain types of populators where the sort order or subset of items may be defined by the end user. The class CrlFilter preferably includes public definitions such as "enum rlFilterType" and "enum rlFrameType". This class also preferably includes public methods such as "CrlFilter," ".about.CrlFilter," "CreatePopulator," "GetName," "GetFilterType," "GetFrameTyp," "GetRoworder," "GetColumnOrder," "GetColumnName," "GetColumnWidth," "GetCriteria," "SetName," "SetFilterType," "SetFrameType," "SetRoworder," "SetColumnOrder" and "SetCriteria."

Detailed Description Text (484):
The class CrlKey is from the Record List module and preferably provides an encapsulation of the information needed to build a record. With each row which is retrieved from the database, the necessary information to completely build the record represented by that row is collected in this class. This class then maintains the two classes required by the record builders, CdaxAccessorKey and CrfRecordClass. The information stored in this class is also used by the populators to correctly build the recordsets with the correct database and database key. This class preferably includes public methods such as "CrlKey," ".about.CrlKey," "GetAccessorKey" and "GetRecordClass."

CLAIMS:

1. A clinical information reporting system for use in coordinating cardiology related patient information, said system including a platform hardware component and an application software component and said system further comprising:

A data input means for the receipt of physiological signals acquired from a patient;

a patient demographic information input means for the receipt of the demographic information of a patient;

said application software component of said system including object oriented software modules having a plurality of tiers including a top tier, a second tier and a third tier with at least one interface for communication between said top tier and said second tier and at least one further interface for communication between said second tier and said third tier;

a database for the receipt of said physiological signals received from said data input means and said demographic information from said demographic information input means in said third tier;

means for arranging said information and said signals in said database to correspond to records containing the said demographic information of the patient for which said physiological signals were acquired in said second tier; and

report generating means for the generation of reports relating to said physiological signals and said demographic information of the patient in said top tier.

3. The clinical information reporting system of claim 2 wherein said application software component includes a dynamically loadable cardiology information system applet for resting ECG interpretation therein.

4. The clinical information reporting system of claim 3 wherein said cardiology information system applet includes analysis software to evaluate said physiological signals generated from a resting ECG report module for a patient to assist the user with the diagnosis of a patient.

5. The clinical information reporting system of claim 2 wherein said application software component includes a dynamically loadable cardiology information system applet for stress testing ECG interpretation therein.

6. The clinical information reporting system of claim 5 wherein said cardiology information system applet includes analysis software to evaluate said physiological signals generated from a stress ECG report module for a patient to assist the user with the diagnosis of a patient.

10. The clinical information reporting system of claim 2 wherein said application software component includes a patient demographics module therein and said patient demographics module is a dynamically loadable applet which performs the patient demographics related functions of said application software component to enable the user of said system to create a demographic report containing said demographic information of the patients in said database.

11. The clinical information reporting system of claim 2 wherein said application software component includes a print/fax/preview module therein and said print/fax/preview module is a dynamically loadable applet which performs the record receipt, transmission and viewing functions of said application software component

to enable the user of said system to print and fax said physiological signals and said demographic information from said database.

13. A clinical information reporting system for use in coordinating cardiology related patient information, said system including a platform hardware component and an application software component and said system further comprising:

A data input means for the receipt of physiological signals acquired from a patient;

a patient demographic information input means for the receipt of demographic information of a patient;

said application software component of said system including object oriented software modules having a plurality of tiers including a top tier a second tier and a third tier with at least one interface for communication between said top tier and said second tier and at least one further interface for communication between said second tier and said third tier;

said application software component further including an object oriented database for the receipt of said physiological signals from said data input means and said patient demographic information from said patient demographic information input means

means for arranging said database to correlate said patient demographic information and said physiological signals of the patient for which physiological signals were acquired;

a framework shell module having an applet interface therein for communication between said top tier and said second tier; and

a dynamically loadable cardiology information system applet in one of said tiers and having further modules therein to perform resting ECG analysis and stress testing ECG analysis on said physiological signals.

14. A method of processing cardiology related information for use in a clinical information reporting system the method including the steps of;

providing an application software component including object oriented software modules having a plurality of tiers including a top tier, a second tier and a third tier with at least one interface for communication between said top tier and said second tier and at least one further interface for communication between said second tier and said third tier; acquiring as input, physiological signals from a patient;

storing the physiological signals from a patient in a database having demographic information for the patient wherein the database is located in the third tier of the application software component;

storing procedural information for medical procedures in the third tier of the application software component;

storing administrative information for a health care facility in the third tier of the application software component;

storing record workflow requirements for processing reports generated by the system in the top tier of the application software component; and

generating reports for viewing, editing and/or printing based on the combined data related to the physiological signals of the patient, procedural information and

administrative information in accordance with the record workflow requirements of the system in the second tier of the application software component.

Previous Doc Next Doc Go to Doc#

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L10 and (test near data)

Display: 50 **Documents in Display Format:** FRO **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Friday, December 09, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L12	L10 and (test near data)	2	L12
L11	L10 and (healthcare near provider)	0	L11
L10	l9 and (patient near information)	12	L10
L9	L8 and database	14	L9
L8	L7 and (limit$ near access)	14	L8
L7	cardiolog$ and healthcare	206	L7
L6	cardiolog$ near healthcare	0	L6
L5	cariolog$ near healthcare	0	L5
L4	cardiol$ near healthcare	0	L4
L3	l1 and (patient near test near data)	1	L3
L2	L1 and (limit$ near access)	1	L2
L1	(patient near test) and (patient near information) and (healthcare near provider)	16	L1

END OF SEARCH HISTORY

L7: Entry 3 of 11 File: PGPB Jan 6, 2005

DOCUMENT-IDENTIFIER: US 20050004895 A1
TITLE: System and method for implementing a global master patient index

Summary of Invention Paragraph:
[0005] Duplicate patient records create problems for hospitals, physician practices, and other parties involved in healthcare. Failure to find preexisting records creates duplicate files for the same individual. Duplicate records splinter clinical information and hinder access. Impaired access to complete patient information hinders a clinician's ability to quickly and accurately treat a patient. Duplicate records impact billing processes and decision support for administration.

Brief Description of Drawings Paragraph:
[0040] FIGS. 11-107 describe an exemplary laboratory application operable to use a GMPI to integrate patient information across healthcare businesses.

Detail Description Paragraph:
[0047] FIG. 1 is a block diagram illustrating the use of a Global Master Patient Index, or GMPI, as enabled by one embodiment of the present invention. Several exemplary sites 60 are shown. Each site 60 may be associated with a health care organization, facility, or business, such as a physician's office, laboratory, health plan, hospital, etc. The individual sites 60 may be operable to share various types of information with each other, including patient record information, such as patient contact and billing information, patient medical history, etc. In other words, the sites 60 may be associated with a Health Data Network (HDN), and the organization or business associated with each site may be referred to as an HDN Business. It is noted that the sites 60 shown in FIG. 1 represent typical types of businesses found in a typical Health Data Network, and any of various other organizations may be present in other embodiments of a Health Data Network. Also, any number of organizations or businesses may be connected to the HDN.

Detail Description Paragraph:
[0048] As shown, each site 60 may utilize a computer system 62 that interfaces to one or more databases 64. Among other types of information, a database 64 may store patient record information The use of patient record information may differ for the various sites. For example, Physician's Office A (site 60B) may primarily use the patient records to view and update clinical information regarding patients' medical history, while the Health Plan (site 60D) may primarily use the patient records to manage insurance information for the respective patients.

Detail Description Paragraph:
[0062] FIG. 2 also illustrates an object dictionary 120. The client object server 110 interfaces with the object dictionary 120 to dynamically determine the data location(s), layout, and retrieval/storage methods. The object dictionary comprises metadata information regarding the data location(s), layout, and retrieval/storage methods for each object that client applications may request from the client object server. For example, the object dictionary may comprise information regarding a customer invoice object, as in the example above. The types of objects that may be defined and managed by the client object server is of course unlimited, and may depend on the purpose of a particular system or application. For example, a

healthcare system may define objects representing patients, healthcare providers, etc. The object definitions may be dynamically changed by changing the object dictionary metadata information.

Detail Description Paragraph:
[0129] In step 412, user input requesting to confirm the link is received. For example, as described above, when the user requests a patient record with unconfirmed links to be displayed, a user interface enabling the user to specify confirmation information for the patient record may be displayed, and the user may request to confirm an unconfirmed link via this user interface. Also, a user may confirm a link between A and B that was previously denigrated.

Detail Description Paragraph:
[0190] In step 304, the record for the specified patient is received, and the record information is used to populate patient information fields of the Requisition window. In one embodiment, the system may be operable to maintain a Global Master Patient Index (GMPI) that integrates patient record information for multiple Health Data Network Businesses. Thus, this GMPI information may be used in retrieving the appropriate patient record.

Detail Description Paragraph:
[0191] In step 306, user input specifying general requisition information is received, such as contact information for the patient, guarantor information, etc. FIG. 15 illustrates an exemplary user interface for receiving this general information, i.e., the General page of the Requisition window displayed in step 300.

Detail Description Paragraph:
[0193] In step 308, user input specifying billing information for the requisition is received. FIG. 18 illustrates an exemplary user interface for receiving this billing information, i.e., the Billing page of the Requisition window displayed in step 300. In one embodiment, when the user moves from the General page to another page, such as the Billing page, any data the user has entered in the patient information fields is automatically saved in the patient's record. A message may appear, advising the user that all requisitions will now use the new patient information. In one embodiment, the user may be able to choose whether or not to modify the patient record in this way. It is noted that the fields included in the user interface that is displayed in step 308 may depend on the Bill Type chosen by the user.

Detail Description Paragraph:
[0210] A bill type of Patient means that the patient will be billed for services rendered. The user will need to enter guarantor information for the patient on the Billing page.

Detail Description Paragraph:
[0214] For various of the above fields, if the user selects an existing patient and the information exists in their record, the field may be automatically populated. Changes made to the field may also change the patient's existing record.

Detail Description Paragraph:
[0238] The Additional Info page (see FIG. 20) includes fields for entering additional information regarding the requisition. The test codes the user selects on the Test Codes page, the Bill Type setting on the General page and the type of the active HDN Business (e.g., "physician practice" or "hospital") determines which of these fields is required For example, when the user orders a test that requires 24 hour urine samples, the user is asked a series of questions such as the patient's height, weight and urine volume. In this case, the user would complete these fields: Lab Instructions, Urine Volume ML, and Urine Hrs, and any other information relevant to the patient or test being performed.

Detail Description Paragraph:
[0263] The user can find requisitions by using requisition information or by using patient information. Each method enables the user to use different parameters to narrow down the results of the user's search. For example, the user may want to generate a list of all the entered requisitions whose specimens were collected within a certain time period, or the user may want to obtain a list of all the requisitions that were ordered by a physician for a patient. Also, there may be times when the user needs to add more information to an existing requisition that has not been transmitted to a lab yet, as in the case where a doctor requests an additional test for a patient or the user needs to change information on the patient's insurance coverage. In both cases, the user would search for the requisition, make the required changes to it and then save it. A doctor may also decide to cancel a requisition, in which case, the user would delete that requisition because it is no longer needed.

Detail Description Paragraph:
[0506] Patient information is used by many other modules in the system and is shared within the user's organization as well as other organizations participating in the care of the patient. Therefore, great care should be taken to maintain the accuracy and integrity of this information. The system includes various features for helping to maintain data integrity, as described below.

Detail Description Paragraph:
[0508] Enter and modify demographic information in a patient record

Detail Description Paragraph:
[0512] Add, modify and remove guarantor information in a patient record

Detail Description Paragraph:
[0514] Add, modify and remove insurance information in a patient record

Detail Description Paragraph:
[0517] Add, modify and remove consent information in a patient record

Detail Description Paragraph:
[0531] The Employment page of the Patient Details window (not shown) lists employment information for the patient, both past and present, and includes employer name, address, phone numbers, employment period and position. This page may be used to edit or enter new employment information for the patient.

Detail Description Paragraph:
[0533] The Guarantors page of the Patient Details window (not shown) lists the person(s) responsible for payment for any medical procedures not covered by a payer or a third party. A guarantor can be the patient, a parent/guardian, the patient's spouse, the patient's employer, or any other person financially responsible for the patient's medical expenses. This page may be used to edit or enter guarantor information for the patient.

Detail Description Paragraph:
[0537] The Insurance page of the Patient Details window (not shown) lists insurance information, both current and expired, for the patient, and includes insurance code, payer, insured name, policy/member number and effective dates. This page may be used to edit or enter insurance information for the patient.

Detail Description Paragraph:
[0541] The Contacts page of the Patient Details window (not shown) lists persons who are contacts for the patient, and includes the contact's name, address, phone numbers and relationship to the patient. This page may be used to edit or enter contact information for the patient.

Detail Description Paragraph:
[0543] The Consent page of the Patient Details window (not shown) indicates whether there is a valid patient consent form on file for a particular patient record. This page may be used to edit or enter consent information for the patient.

Detail Description Paragraph:
[0545] The Orders page of the Patient Details window (not shown) lists all laboratory requisitions that have been prepared for the patient. To create a new standard requisition, the user can click Create New. The Requisition window appears with the General page active and the patient information populating the fields. This page may also be used to edit order information for the patient.

Detail Description Paragraph:
[0549] The Diagnosis Codes page of the Patient Details window (not shown) lists diagnosis codes which have been associated with the patient, either manually through this page or automatically when a requisition is created for the patient. This page may be used to edit or enter diagnosis code information for the patient.

Detail Description Paragraph:
[0551] Because patient records are setup and maintained by multiple users at multiple facilities in the Health Data Network, it is possible that a patient will have multiple patient records. This can create problems when determining which record to maintain. Duplicate records can splinter clinical data and hinder access to patient information.

Detail Description Paragraph:
[0552] For this reason, the system implements a Global Master Patient Index (GMPI). The GMPI can link multiple records together for the same patient. Thus, the GMPI gathers patient information together under a single umbrella. In the preferred embodiment, GMPI functionality within the system includes:

Detail Description Paragraph:
[0622] The system provides the ability to secure information across a large and open network of computers and the people that use them. This network is referred to herein as a Health Data Network, or HDN. The security of this network, including access to it, is critical because the system provides access to confidential patient information, including laboratory test results and medical history.

Detail Description Paragraph:
[0672] Entering Insurance Information for a Patient

Detail Description Paragraph:
[0804] FIG. 93 illustrates the Specialties page of the Provider Details window. A specialty defines a specific area of medicine, such as cardiology, pediatrics, or oncology, that a provider supplies. On the Specialties page of the Providers subsystem, the user can view specialties associated with the selected Provider. Each specialty record includes a description, and the specialties are linked to caregivers. Management of provider specialties is carried out through the Specialties page of the Provider Details window.

Detail Description Paragraph:
[0809] Security administration involves setting up and maintaining security aspects of the system. Using the Manage Security submenu of the Admin menu, the user, i.e., an administrator, can control user access to confidential patient information stored on the network. For example, the security features may prevent a data record from being viewed or modified by unauthorized users.

Detail Description Paragraph:
[0813] Roles and Permissions--In the preferred embodiment, the security system is

based on roles and permissions. These two concepts, combined with ownership, determine a user's authorization level, or the operations the user can carry out. A role is a group of operations that is typically related to a specific function or position. For example, various roles may be defined for physicians, nurses, office assistants, etc., or for any other function or position that a business desires. Roles limit transaction access to certain groups of users. For example, roles can be used to deny access to transactions related to clinical results except for people whose job requires that they have access. Only people with an approved need to know should be assigned roles that have search and read capabilities on patient information. The system users are classified and their permissions are assigned based on pre-defined security roles.

Detail Description Paragraph:
[0845] There are preferably no limitations on the number of permission profiles that can be assigned to users. In one embodiment, permission profiles are limited to one role each, and more than one permission profile may be assigned to a user who has several roles. Also, the same permission profile may be assigned to different users who perform the same role. For example, three different hospital registration clerks could share the same role that allows them to view and modify patient information.

Detail Description Paragraph:
[0853] As descried above, a role comprises a group of operations that is typically related to a specific function or position. Roles limit transaction access to certain groups of users. For example, roles can be used to deny access to transactions related to clinical results except for people whose job requires that they have access to this information. Only people with an approved need to know should be assigned roles that have search and read capabilities on patient information.

Detail Description Paragraph:
[0898] The user may define and maintain the user's own code sets, such as groups of values or symbols used to represent information such as a patient's employment status, religion, marital status, etc. These values usually appear in drop-down lists from which the user makes a selection. To handle the code sets, the system may be operable to map and translate global and local codes. Global codes refer to user-defined codes that are used uniformly across the entire network (hub). Local codes refer to user-defined codes that are specific to a certain HDN Business.

Previous Doc Next Doc Go to Doc#

L7: Entry 8 of 11 File: PGPB Jun 5, 2003

DOCUMENT-IDENTIFIER: US 20030105648 A1
TITLE: Integrated insurance eligibility service for an electronic laboratory application

Abstract Paragraph:
A system and method for connecting a healthcare business to a plurality of laboratories. The system includes a client application which executes on a workstation, e.g., at a physician's office or other business. Using this application, a user may place a laboratory order (requisition) for a patient. User input specifying general requisition information is received, such as contact information for the patient, guarantor information, billing information, etc. After the requisition information has been entered, the requisition may be validated by the system. If there are errors in the information entered for the requisition, an error message may appear, and the user may be required to correct the errors. As part of performing this validation, the system may verify insurance information for the requisition, if applicable. In one embodiment, the system interfaces with a separate eligibility server to check for valid patient insurance. For example, the eligibility server may receive the patient name, social security number, and/or other identification information and use this information to check the patient's insurance status. If the patient does not have insurance coverage for the requested lab tests to be performed, the insurance information may be corrected, or other means for payment may be required. Once the requisition information has been validated, the information may be stored. The user may later use a menu option to select and electronically send the requisitions, e.g., by interfacing a middleware server which interfaces with the system for each laboratory which receives the requisitions. In another embodiment, the patient insurance information may be verified after being sent to the middleware server. The middleware server may interface with a separate eligibility server to verify the insurance information. If the insurance information is verified then the middleware server may transmit the requisition information to one or more laboratories specified in the requisition. Otherwise, the originating caregiver may be notified, and the requisition information may not be sent to the laboratories.

Summary of Invention Paragraph:
[0006] FIGS. 1A-1B are a flowchart diagram illustrating a typical workflow among various parties, as known in the prior art, when performing one or more laboratory tests for a patient. Several disadvantages are associated with such a workflow. To name a few:

Summary of Invention Paragraph:
[0020] User input specifying general requisition information is received, such as contact information for the patient, guarantor information, etc. Billing information may also be received. If a requisition was previously created for the specified patient, relative information from that requisition may populate the appropriate user interface fields.

Brief Description of Drawings Paragraph:
[0026] FIGS. 1A-1B (prior art) are a flowchart diagram illustrating a typical workflow among various parties, as known in the prior art, when performing one or more laboratory tests for a patient;

Detail Description Paragraph:
[0040] As shown, each site 60 may utilize a computer system 62 and may also utilize one or more databases 64. Among other types of information, a database 64 may store patient record information. The use of patient information may differ for the various sites. For example, Physician's Office A (site 60B) may primarily use the patient records to view and update clinical information regarding patients' medical history, while the Health Plan (site 60D) may primarily use the patient records to manage insurance information for the respective patients.

Detail Description Paragraph:
[0051] FIG. 3 also illustrates an object dictionary 120. The client object server 110 interfaces with the object dictionary 120 to dynamically determine the data location(s), layout, and retrieval/storage methods. The object dictionary comprises metadata information regarding the data location(s), layout, and retrieval/storage methods for each object that client applications may request from the client object server. For example, the object dictionary may comprise information regarding a customer invoice object, as in the example above. The types of objects that may be defined and managed by the client object server is of course unlimited, and may depend on the purpose of a particular system or application. For example, a healthcare system may define objects representing patients, healthcare providers, etc. The object definitions may be dynamically changed by changing the object dictionary metadata information.

Detail Description Paragraph:
[0059] FIG. 5 is a flowchart diagram illustrating one embodiment of a method for electronically sending lab requisitions to laboratories and electronically receiving lab results. In step 200, a user requests to initiate a requisition. An exemplary user interface for entering lab requisition information is described below. In step 202, user information specifying the requisition is received, such as patient information, billing information, test code information, etc.

Detail Description Paragraph:
[0103] In step 304, the record for the specified patient is received, and the record information is used to populate patient information fields of the Requisition window. In one embodiment, the system may be operable to maintain a Global Master Patient Index (GMPI) that integrates patient record information for multiple Health Data Network Businesses. Thus, this GMPI information may be used in retrieving the appropriate patient record.

Detail Description Paragraph:
[0104] In step 306, user input specifying general requisition information is received, such as contact information for the patient, guarantor information, etc. FIG. 10 illustrates an exemplary user interface for receiving this general information, i.e., the General page of the Requisition window displayed in step 300.

Detail Description Paragraph:
[0106] In step 308, user input specifying billing information for the requisition is received. FIG. 13 illustrates an exemplary user interface for receiving this billing information, i.e., the Billing page of the Requisition window displayed in step 300. In one embodiment, when the user moves from the General page to another page, such as the Billing page, any data the user has entered in the patient information fields is automatically saved in the patient's record. A message may appear, advising the user that all requisitions will now use the new patient information. In one embodiment, the user may be able to choose whether or not to modify the patient record in this way. It is noted that the fields included in the user interface that is displayed in step 308 may depend on the Bill Type chosen by the user.

Detail Description Paragraph:
[0123] A bill type of Patient means that the patient will be billed for services rendered. The user will need to enter guarantor information for the patient on the Billing page.

Detail Description Paragraph:
[0127] For various of the above fields, if the user selects an existing patient and the information exists in their record, the field may be automatically populated. Changes made to the field may also change the patient's existing record.

Detail Description Paragraph:
[0148] The Additional Info page (see FIG. 15) includes fields for entering additional information regarding the requisition. The test codes the user selects on the Test Codes page, the Bill Type setting on the General page and the type of the active HDN Business (e.g., "physician practice" or "hospital") determines which of these fields is required. For example, when the user orders a test that requires 24 hour urine samples, the user is asked a series of questions such as the patient's height, weight and urine volume. In this case, the user would complete these fields: Lab Instructions, Urine Volume ML, and Urine Hrs, and any other information relevant to the patient or test being performed.

Detail Description Paragraph:
[0173] The user can find requisitions by using requisition information or by using patient information. Each method enables the user to use different parameters to narrow down the results of the user's search. For example, the user may want to generate a list of all the entered requisitions whose specimens were collected within a certain time period, or the user may want to obtain a list of all the requisitions that were ordered by a physician for a patient. Also, there may be times when the user needs to add more information to an existing requisition that has not been transmitted to a lab yet, as in the case where a doctor requests an additional test for a patient or the user needs to change information on the patient's insurance coverage. In both cases, the user would search for the requisition, make the required changes to it and then save it. A doctor may also decide to cancel a requisition, in which case, the user would delete that requisition because it is no longer needed.

Detail Description Paragraph:
[0416] Patient information is used by many other modules in the system and is shared within the user's organization as well as other organizations participating in the care of the patient. Therefore, great care should be taken to maintain the accuracy and integrity of this information. The system includes various features for helping to maintain data integrity, as described below.

Detail Description Paragraph:
[0418] Enter and modify demographic information in a patient record

Detail Description Paragraph:
[0422] Add, modify and remove guarantor information in a patient record

Detail Description Paragraph:
[0424] Add, modify and remove insurance information in a patient record

Detail Description Paragraph:
[0427] Add, modify and remove consent information in a patient record

Detail Description Paragraph:
[0441] The Employment page of the Patient Details window (not shown) lists employment information for the patient, both past and present, and includes employer name, address, phone numbers, employment period and position. This page may be used to edit or enter new employment information for the patient.

Detail Description Paragraph:
[0443] The Guarantors page of the Patient Details window (not shown) lists the person(s) responsible for payment for any medical procedures not covered by a payer or a third party. A guarantor can be the patient, a parent/guardian, the patient's spouse, the patient's employer, or any other person financially responsible for the patient's medical expenses. This page may be used to edit or enter guarantor information for the patient.

Detail Description Paragraph:
[0447] The Insurance page of the Patient Details window (not shown) lists insurance information, both current and expired, for the patient, and includes insurance code, payer, insured name, policy/member number and effective dates. This page may be used to edit or enter insurance information for the patient.

Detail Description Paragraph:
[0451] The Contacts page of the Patient Details window (not shown) lists persons who are contacts for the patient, and includes the contact's name, address, phone numbers and relationship to the patient. This page may be used to edit or enter contact information for the patient.

Detail Description Paragraph:
[0453] The Consent page of the Patient Details window (not shown) indicates whether there is a valid patient consent form on file for a particular patient record. This page may be used to edit or enter consent information for the patient.

Detail Description Paragraph:
[0455] The Orders page of the Patient Details window (not shown) lists all laboratory requisitions that have been prepared for the patient. To create a new standard requisition, the user can click Create New. The Requisition window appears with the General page active and the patient information populating the fields. This page may also be used to edit order information for the patient.

Detail Description Paragraph:
[0459] The Diagnosis Codes page of the Patient Details window (not shown) lists diagnosis codes which have been associated with the patient, either manually through this page or automatically when a requisition is created for the patient. This page may be used to edit or enter diagnosis code information for the patient.

Detail Description Paragraph:
[0461] Because patient records are setup and maintained by multiple users at multiple facilities in the Health Data Network, it is possible that a patient will have multiple patient records. This can create problems when determining which record to maintain. Duplicate records can splinter clinical data and hinder access to patient information.

Detail Description Paragraph:
[0462] For this reason, the system implements a Global Master Patient Index (GMPI). The GMPI can link multiple records together for the same patient. Thus, the GMPI gathers patient information together under a single umbrella. In the preferred embodiment, GMPI functionality within the system includes:

Detail Description Paragraph:
[0533] The system provides the ability to secure information across a large and open network of computers and the people that use them. This network is referred to herein as a Health Data Network, or HDN. The security of this network, including access to it, is critical because the system provides access to confidential patient information, including laboratory test results and medical history.

Detail Description Paragraph:

[0583] Entering Insurance Information for a Patient

Detail Description Paragraph:
[0715] FIG. 88 illustrates the Specialties page of the Provider Details window. A specialty defines a specific area of medicine, such as cardiology, pediatrics, or oncology, that a provider supplies. On the Specialties page of the Providers subsystem, the user can view specialties associated with the selected Provider. Each specialty record includes a description, and the specialties are linked to caregivers. Management of provider specialties is carried out through the Specialties page of the Provider Details window.

Detail Description Paragraph:
[0720] Security administration involves setting up and maintaining security aspects of the system. Using the Manage Security submenu of the Admin menu, the user, i.e., an administrator, can control user access to confidential patient information stored on the network. For example, the security features may prevent a data record from being viewed or modified by unauthorized users.

Detail Description Paragraph:
[0724] Roles and Permissions--In the preferred embodiment, the security system is based on roles and permissions. These two concepts, combined with ownership, determine a user's authorization level, or the operations the user can carry out. A role is a group of operations that is typically related to a specific function or position. For example, various roles may be defined for physicians, nurses, office assistants, etc., or for any other function or position that a business desires. Roles limit transaction access to certain groups of users. For example, roles can be used to deny access to transactions related to clinical results except for people whose job requires that they have access. Only people with an approved need to know should be assigned roles that have search and read capabilities on patient information. The system users are classified and their permissions are assigned based on pre-defined security roles.

Detail Description Paragraph:
[0756] There are preferably no limitations on the number of permission profiles that can be assigned to users. In one embodiment, permission profiles are limited to one role each, and more than one permission profile may be assigned to a user who has several roles. Also, the same permission profile may be assigned to different users who perform the same role. For example, three different hospital registration clerks could share the same role that allows them to view and modify patient information.

Detail Description Paragraph:
[0764] As described above, a role comprises a group of operations that is typically related to a specific function or position. Roles limit transaction access to certain groups of users. For example, roles can be used to deny access to transactions related to clinical results except for people whose job requires that they have access to this information. Only people with an approved need to know should be assigned roles that have search and read capabilities on patient information.

Detail Description Paragraph:
[0809] The user may define and maintain the user's own code sets, such as groups of values or symbols used to represent information such as a patient's employment status, religion, marital status, etc. These values usually appear in drop-down lists from which the user makes a selection. To handle the code sets, the system may be operable to map and translate global and local codes. Global codes refer to user-defined codes that are used uniformly across the entire network (hub). Local codes refer to user-defined codes that are specific to a certain HDN Business.

Previous Doc Next Doc Go to Doc#